VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Uživatelsky přívětivé sestavování dotazů nad relační databází

User-Friendly Settings of Queries over Relational Database

2018 Martina Vápeníková

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Martina Vápeníková**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Uživatelsky přívětivé sestavování dotazů nad relační databází
User-Friendly Settings of Queries over Relational Database

Jazyk vypracování: čeština

Zásady pro vypracování:

Na Katedře informatiky je vyvíjen informační systém pro analýzu dat poruch v elektrických sítích, který je využíván distributory elektrické energie v České i Slovenské republice. Cílem této práce je návrh a implementace formulářů pro uživatelsky přívětivé sestavování dotazů nad touto databází.

1. Nastudujte stávající informační systém a přístupy pro sestavování uživatelských dotazů.
2. Navrhněte a naimplementujte formuláře pro uživatelsky přívětivé sestavování dotazů nad touto databází.
3. Vytvořené řešení vyhodnoťte a otestujte.

Seznam doporučené odborné literatury:

Podle pokynů vedoucího bakalářské práce.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **doc. Ing. Michal Krátký, Ph.D.**

Datum zadání: 01.09.2017

Datum odevzdání: 30.04.2018

doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracovala samostatně. Uvedla jsem všechny literární prameny a publikace, ze kterých jsem čerpala.

V Ostravě 22. dubna 2018 ..... Šafeníková .....

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.

V Ostravě 22. dubna 2018 ..... Fajeníková .....

Ráda bych na tomto místě poděkovala vedoucímu bakalářské práce panu doc. Ing. Michalovi Krátkému, Ph.D. za věcné připomínky k obsahu, protože bez něj by tato práce nevznikla. Zároveň bych také chtěla poděkovat rodině a příteli, kteří mi pomohli tuto práci zdokonalit.

Abstrakt

Tématem této bakalářské práce je návrh a implementace formulářů, které uživateli usnadní sestavování dotazů nad relační databází. Tyto formuláře jsou navrženy pro již existující informační systém týkající se analýzy dat poruch v elektrických sítích. Součástí řešení je také volba vhodného dotazovacího jazyka pro vytvářenou aplikaci, která bude obecná a použitelná pro databáze se schématem typu hvězda. Zhotovená aplikace bude popsána z hlediska analýzy jednotlivých funkcí systému, návrhu uživatelského rozhraní a samotné implementace.

Klíčová slova: informační systém, dotazovací jazyky, relační datový model, návrh uživatelského rozhraní

Abstract

The theme of this thesis is design and implementation of forms which make it easier for user to settings of queries over relational database. These forms are designed for already existing information system related to analysis of fault data in electrical networks. The part of the solution is also the choice of suitable query language for the created application, which will be general and usable for databases based on star schema. The created application will be describe in terms of analysis of separate functions of system, user interface design and implementation itself.

Key Words: information system, query languages, relational model of data, user interface design

Obsah

Seznam použitých zkratek a symbolů

DB	–	Databáze
SŘBD	–	Systém řízení bází dat
DBS	–	Databázový systém
IBM	–	International Business Machines Corporation
DML	–	Data Manipulation Language
DDL	–	Data Definition Language
DCL	–	Data Control Language
SQL	–	Structured Query Language
SEQUEL	–	Structured English Query Language
SQUARE	–	Specifying Queries As Relational Expressions
QUEL	–	QUEry Language
INGRES	–	INteractive Graphics and REtrieval System
PIQUE	–	Pits QUEry language
PITS	–	Pie-In-The-Sky
ISBL	–	Information System Base Language
PRTV	–	Peterlee Relational Test Vehicle
VQLs	–	Visual Query Languages
VQSs	–	Visual Query Systems
QBE	–	Query By Example
QBD	–	Query By Diagram
MashQL	–	Mashup Query Language
SPARQL	–	Simple Protocol and RDF Query Language
W3C	–	World Wide Web Consortium
RDF	–	Resource Description Framework - systém popisu zdrojů
URI	–	Uniform Resource Identifier – jednotný identifikátor zdroje
IS	–	Informační Systém
C#	–	Vysokoúrovňový objektově orientovaný programovací jazyk
ASP.NET	–	Součást .NET Frameworku pro tvorbu webových aplikací a služeb
HTTP	–	Hypertext Transfer Protocol

Seznam obrázků

Seznam tabulek

1 Úvod

Tato bakalářská práce zahrnuje návrh a implementaci formulářů pro již existující informační systém zabývající se analýzou dat poruch v elektrických sítích. Tento informační systém je hojně používaný distributory elektrické energie nejen v České, ale i Slovenské Republice. Formuláře budou vytvořeny takovým způsobem, aby s nimi mohli pracovat také uživatelé s laickými znalostmi databázových systémů. Neméně důležité je splnit současně podmínku týkající se spokojenosti uživatelů při práci s vyvíjenými formuláři.

Pro naplnění požadavků definovaných výše je tedy podstatné nastudovat problematiku dotazovacích jazyků a následně provést jejich porovnání mezi sebou s ohledem na syntaxi a názor uživatele při práci s těmito jazyky. Po nalezení vhodného jazyka lze začít s návrhem samotného řešení zahrnujícího popis funkcí, návrh uživatelského rozhraní a samotnou implementaci vytvořené aplikace. Jako součást implementace bude prezentován třídní diagram s přesným popisem jednotlivých komponent daného systému. Na závěr se implementované řešení otestuje nad jinou databází, kdy prvním krokem bude určení vhodného schématu databáze, se kterým bude schopna vyvinutá aplikace pracovat. Řešení bude tedy obecné a vhodné pro databáze se schématem typu hvězda. Bakalářská práce je strukturovaná do čtyř částí. Kapitola 2 se zabývá popisem dotazovacích jazyků včetně srovnání jejich syntaxí a také přívětivosti k uživateli. Závěrem této kapitoly je volba vhodného dotazovacího jazyka, který je pak použit pro implementaci vyvíjené aplikace. Kapitola 3 se věnuje analýze vyvíjené aplikace, která je zobecněna pro databáze s určitým typem schématu. Následně jsou popsány funkce dané aplikace. Kapitola 4 řeší návrh aplikace, tedy vstupní a výstupní data spolu s uživatelským rozhraním. Kapitola 5 se věnuje samotné implementaci aplikace, kde jsou představeny použité technologie a popsány komponenty třídního diagramu. V poslední kapitole 6 je znázorněna práce s vyvinutou aplikací.

2 Databázové systémy

2.1 Úvod

V minulosti byly důležité informace potřebné pro společnost uchovávány či zaznamenávány na papírech a archivovány jako papírové kartotéky. Při změnách informací bylo tedy nutné nejprve nalézt požadovaná data a poté je aktualizovat. Tento proces trval poměrně dlouhou dobu, avšak vznikem a rozvojem počítačů byl ve většině případech nahrazen uložením těchto informací do tzv. *databáze*. Existuje mnoho typů databází, nicméně s ohledem na obsah této práce není nutné rozebírat je všechny, z důvodu čehož je níže popsána pouze relační databáze, která bude dále využívána. Jelikož bylo nutné s daty uloženými v databázi pracovat, došlo k vývoji speciálního programového vybavení nazývaného *systém řízení bází dat* (SŘBD). Tento systém ve spojení s databází utváří *databázový systém* (DBS) [1].

Pro přiblížení uspořádání dat v databázích, je nutné zmínit *E-R konceptuální model* nebo-li *E-R model* [2, 3]. Tento model je považován za velice důležitý při vývoji databázového systému, neboť graficky popisuje strukturu databáze na základě skutečných objektů reálného světa a vztahů mezi nimi. Proto lze říci, že E-R model reprezentuje schéma databáze bez zaměření na konkrétní typ databáze. E-R model tedy mapuje množiny objektů stejného typu jako tzv. *entitní typy*, mezi nimiž definuje vazby - tzv. *vztahy*. Pokud mluvíme v rámci tohoto modelu o určitém objektu entitního typu, potom jej označujeme jako tzv. *entita*. Jelikož každý objekt reálného světa je možné jednoznačně určit, každá entita musí být v rámci entitního typu taktéž odlišitelná od ostatních entit, tedy jednoznačně identifikovatelná pomocí tzv. *identifikačního klíče*.

Zatím byly popsány pouze obecné pojmy týkající se databází, avšak je nutné také specifikovat vlastnosti relační databáze. Pojem relační databáze byl definován v letech 1970 americkým vědcem pracujícím v IBM Edgarem F. Coddem v článku "A Relational Model of Data for Large Shared Data Banks" [4]. S tímto typem databáze souvisí také pojem *relační datový model* [2, 3], který určuje způsob, jakým jsou data uložena v databázi. Základem tohoto modelu je tzv. *relace* (dvourozměrná tabulka), do níž jsou uložena data na základě 'hlavičky' relace, která obsahuje jména jednotlivých atributů pro danou entitu. Data jsou tedy ukládána do řádků a sloupců, kde každý řádek představuje hodnoty atributů entity a každý sloupec hodnotu jednoho z těchto atributů. Každý atribut musí mít v rámci relace jednoznačný název a datový typ. Součástí datových typů je samozřejmě také rozsah, který v případě datového typu `VARCHAR` udává délku řetězce. Každý řádek dané relace nazýváme *n-ticí*. V rámci relace je nutné stanovit atribut, na základě kterého je možné odlišit od sebe jednotlivé n-tice. Takový atribut se tedy stává *primárním klíčem*. V případě, kdy je nutné propojit dvě relace mezi sebou na základě jejich atributů, dochází k zavedení pojmu *cizí klíč*. Cizí klíč je tedy také atributem, který odkazuje na primární klíč spojované relace.

Pro práci s relacemi byl vyvinut jazyk vysoké úrovně - *relační algebra* [2, 3]. Jinými slovy,

relační algebra obsahuje množinu operací, jejichž použitím na relaci je vrácena opět relace. Jak již bylo dříve zmíněno, relace jsou množinou n-tic, s čímž souvisí také poskytované operace, kterými jsou průnik, sjednocení, rozdíl a kartézský součin. Základními operacemi relační algebry při dotazování jsou však tyto: *projekce*, *selekce* a *spojení*. Projekce poskytuje pouze výpis hodnot definovaných atributů relace bez jakéhokoliv omezení. Selekce slouží pro výpis pouze těch n-tic relace, které splňují zadanou podmínku. Poslední operace spojení vytváří novou relaci spojením n-tic z obou zadaných relací na základě stejných hodnot atributů ve spojovaných relacích.

Pro získávání informací z databáze se využívají *dotazovací jazyky* [5]. Rozlišujeme dva typy dotazovacích jazyků [6], a to *procedurální jazyky* a *neprocedurální jazyky*. V případě *procedurálních jazyků* je nutné vytvořit algoritmus pro získání požadovaných dat, což znamená přímo definovat, jakým způsobem provést výběr dat. Tyto jazyky jsou tedy vhodné spíše pro profesionální programátory. Na druhé straně *neprocedurální jazyky* jsou snazší, nutné je zadat pouze podmínky pro výběr dat z databáze nebo-li co od databáze požadujeme. V průběhu vývoje dotazovacích jazyků došlo k jejich rozdělení na *textové* a *vizuální* (nazývané také grafické). V dnešní době je však nejznámějším a nejpoužívanějším dotazovacím jazykem SQL [5], který spadá do kategorie textových dotazovacích jazyků a je postaven na relační algebře relačního datového modelu. Dále se zaměříme na jiné dotazovací jazyky, a to jazyky relačních databází, neboť v této práci je kladen důraz na relační databáze. Při práci s relační databází se nabízí široká škála dotazovacích jazyků. Dotazy lze dělit na příkazy pro manipulaci s daty (DML), příkazy pro definici dat (DDL) a příkazy pro řízení dat (DCL).

Pro umožnění srovnávání syntaxí jednotlivých dotazovacích jazyků je v rámci celé práce využit tento datový model s lineárním zápisem:

Legenda: **Tabulka**, primární klíč, *cizí klíč*, atribut

Stavba (id_stavby, typ_stavby, ulice, cislo_popisne, datum_kolaudace)
Dotace_EU (id_dotace, vyse_dotace, datum_prideleni, zpusob_pouziti, *id_stavby*)

2.2 Textové dotazovací jazyky

Použití textových dotazovacích jazyků je tradičním způsobem interakce s databázemi.

2.2.1 SQL

Jazyk SQL [5] je neprocedurálním dotazovacím jazykem, převážně jazykem relačních databází, i přestože jeho využití je širší. Tento jazyk byl vyvinut v roce 1974 zaměstnanci IBM Donaldem D. Chamberlinem a Raymondem F. Boycem pod názvem SEQUEL a byl založen na jejich původním jazyce SQUARE [7, 8], který vytvořili v návaznosti na již zmíněný článek publikovaný Edgarem F. Coddem rovněž pracujícím v IBM. Jazyk SQUARE je z velké části založen na matematické syntaxi zahrnující teorii množin a predikátovou logiku. Přesto poskytl základní prvky

pro manipulaci s databázemi. Později však došlo ke zkrácení názvu SEQUEL na SQL z důvodu již existujícího stejného názvu v oblasti britských leteckých společností.

Pro bližší představu vývoje jazyka SQL je uvedeno srovnání [7, 8] přechodu od syntaxe SQUARE k syntaxi SEQUEL na poměrně jednoduchém příkladu (schéma databáze je popsáno v kapitole 2.1):

Vypsání seznamu všech dotací (`vyse_dotace, datum_prideleni`), které byly použity na rekonstrukci stavby s identifikačním číslem 10 (`id_stavby`).

SQUARE:

```
vyse_dotace, datum_prideleni
Dotace_EU zpusob_pouziti, id_stavby ('rekonstrukce', 10)
```

SEQUEL:

```
SELECT vyse_dotace, datum_prideleni
FROM Dotace_EU
WHERE zpusob_pouziti = 'rekonstrukce'
AND id_stavby = 10
```

Vidíme, že syntaxe dotazovacího jazyka SEQUEL se více podobá angličtině, což je lepší pro uživatele neznalého v oblasti databází, neboť mu umožňuje snazší orientaci při tvorbě dotazu. Z toho důvodu lze říci, že jazyk SEQUEL je přehlednější a srozumitelnější.

Mezi základní vlastnosti jazyka SQL [5] patří:

- Data se v databázi ukládají formou tabulek. Tabulky mohou být dvou typů: skutečné (odpovídající schématu databáze) nebo virtuální (tzv. pohledy).
- SQL zprostředkovává data programu nebo uživateli, jenž nemusí být obeznámen s fyzickou strukturou nebo umístěním dat.
- Umístění tabulek v databázi ani pořadí sloupců v tabulkách není podstatným údajem, protože tabulky i sloupce jsou identifikovány vlastním jménem.
- Pořadí řádků v tabulkách není taktéž podstatné, neboť řádky jsou identifikovány hodnotami ve sloupcích.
- Data mohou být ukládána v různých datových strukturách, avšak programu nebo uživateli jsou neustále prezentována jako tabulky.

2.2.2 Ostatní textové dotazovací jazyky

QUEL [9, 10] je dotazovací jazyk pro relační datový model velice podobný SQL, avšak vznikl jako vlastní jazyk relačního databázového systému INGRES [11]. Na vzniku jazyka se podílel Michael Stonebraker [1], dnes vědec z Massachusettského technologického institutu (MIT). V pozdějších letech byl však nahrazen SQL.

PIQUE [12, 13] je dotazovací jazyk pro relační datový model, který slouží pro vyhledávání dat pro databázový systém PITS [13], což znamená, že PIQUE neobsahuje příkazy pro manipulaci s daty. Má syntaxi podobnou jazyku QUEL.

ISBL [9] je jeden z prvních dotazovacích jazyků pro relační datový model založený na relační algebře. Tento jazyk byl vyvinut v roce 1976 jako dotazovací jazyk pro PRTV systém [11], což byl jeden z prvních databázových systémů implementujících relační datový model E. F. Codda. ISBL nepodporuje příkazy pro manipulaci s daty a neobsahuje žádné agregační funkce (např. průměr). V první řadě je tedy ISBL deklarativním dotazovacím jazykem. Mezi základní operace, které tento jazyk podporuje, patří s ohledem na relační algebru operace sjednocení, rozdíl, průnik, spojení, projekce a selekce. Operátorem používaným v ISBL pro sjednocení je "+", pro rozdíl "-", pro průnik ":", pro spojení "*", pro projekci "%" a pro selekci ":". Obecně existují dva typy výrazů v ISBL [14]:

1. zobrazení výsledku zpracovaného příkazu `List<expression>`
2. přiřazení výsledku zpracovaného výrazu do relace `R=<expression>`

Ve druhém typu výrazu představuje R proměnnou, jejíž hodnotu zastupuje relace.

2.3 Vizuální dotazovací jazyky

Vizuální dotazovací jazyky (tzv. VQLs) [15] se liší od textových dotazovacích jazyků tím, že k zobrazení příslušné oblasti z databáze využívají vizuální reprezentaci, tzn. poskytují jazyk pro vyjádření dotazů ve vizuální formě. VQLs jsou orientovány směrem k širokému spektru uživatelů, především nováčkům, kteří mají limitované počítačové znalosti a nejsou obeznámeni s vnitřní strukturou používané databáze. Je zřejmé, že systémy, které využívají VQLs, označujeme jako Vizuální dotazovací systémy (VQSs).

Impulzem pro vznik VQLs byly následující aspekty:

- Poskytnutí přátelské interakce "člověk-počítač".
- Dovolení neznalému uživateli vyhledání informací v databázích.
- Nalezení mechanismu pro pohodlnou navigaci i v případě neúplných a nejednoznačných dotazů.

[1] https://www.csail.mit.edu/person/michael-stonebraker

Jak již bylo zmíněno, hlavním znakem při práci s VQLs je vizuální reprezentace dotazu, která může být samozřejmě omezena reprezentací dat v databázi, avšak obě zmíněné reprezentace mohou být navzájem odlišné. Ve vizuální reprezentaci se často využívají prvky jako jsou formuláře, diagramy, ikony, popřípadě jejich kombinace. Na základě toho dělíme vizuální reprezentace do 4 typů:

1. Reprezentace založené na formulářích.
2. Reprezentace založené na diagramech.
3. Reprezentace založené na ikonách.
4. Hybridně založené reprezentace.

Z výše uvedených čtyř typů jsou reprezentace tvořené formuláři pro uživatele nejjednodušší, neboť jim přináší přívětivé rozhraní pro manipulaci s daty a z toho důvodu jsou hojně využívány ve spojení s relačními databázemi, přičemž formuláře prezentují tabulky databáze. Samotný dotaz do databáze je pak formulovaný vyplněním daných polí ve formuláři, který je vizualizován uživateli. Předchůdcem VQLs byl dotazovací jazyk QBE [1, 16].

2.3.1 QBE

QBE [1, 16] je prvním grafickým dotazovacím jazykem postaveným na reprezentaci založené na formulářích, o jehož vznik se zasloužil Moshé M. Zloof v IBM Research během 70.let, kdy byl paralelně vyvíjen také jazyk SQL. Základní princip při tvorbě dotazů spočívá ve vyplnění zjednodušených tabulek, které jsou totožné s tabulkami v databázi, což umožňuje jednoduchou práci s databázemi i v případě jejich minimálních znalostí. Jedinou nevýhodou však je, že výsledkem zjednodušení práce s databází je možnost sestavení pouze jednodušších dotazů. Jazyk QBE automaticky odstraňuje duplicitní záznamy, avšak v případě, kdy je nezbytné tyto použít, pak budou vypsány zahrnutím klauzule ALL.

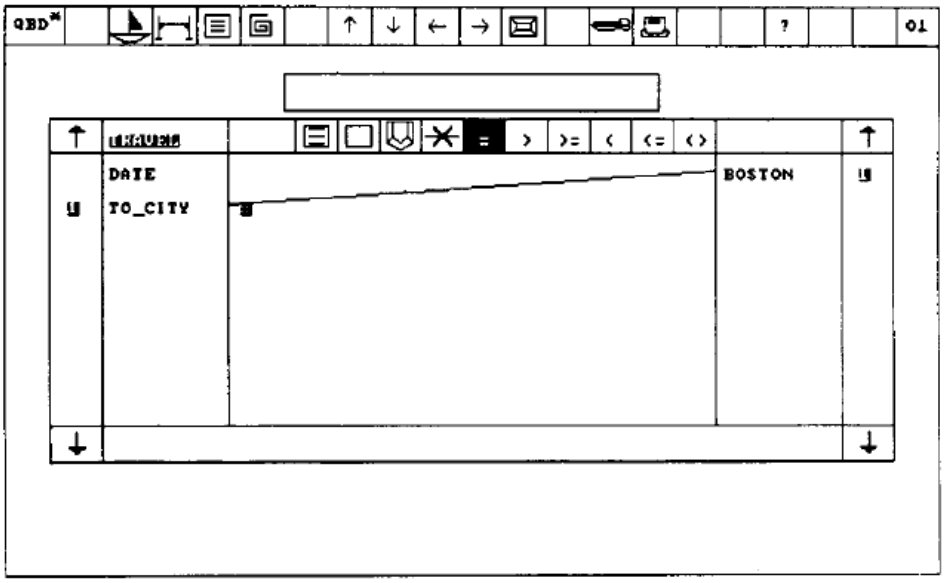
2.3.2 QBD

QBD [17] je grafickým dotazovacím jazykem využívajícím reprezentaci založenou na diagramech. Hlavní princip tohoto jazyka spočívá ve vyjádření základních kroků při formulaci dotazu pomocí skupin grafických operací, jimiž mohou být výběry ikon a konceptů. Pro možné formální stanovení syntaxe i sémantiky daného jazyka, bylo nutno zavést individuální korespondenci mezi grafickými operacemi a syntaktickými konstrukcemi použitými v textových dotazovacích jazycích.

Dotaz v dotazovacím jazyce QBD je obecně strukturován lokalizací významného konceptu označovaného jako hlavní koncept (tj. entita nebo vztah), který lze vnímat také jako vstupní bod pro další poddotazy. Dalším vytvářením již zmíněných poddotazů dochází v rámci schématu

k přesunům od hlavního k ostatním konceptům. Použitím známých operátorů z jiných dotazovacích jazyků (např. sjednocení, průnik, ...) vzniká možnost kombinovat poddotazy mezi sebou. Princip při formulaci dotazu uživatelem v jazyce QBD může probíhat několika způsoby, a to zvolením potřebné ikony v rámci navigace, hlavního konceptu pomocí myši, podmínek atributů hlavní tabulky a cesty následujících konceptů vedoucí od hlavní tabulky. Soupis atributů je však otevřen v novém okně, které obsahuje mimo jiné také ikony potřebné pro specifikaci podmínek.

Pro snazší představu uživatelského rozhraní a postupu při stanovení podmínky daného sloupce tabulky je možné zhlédnout obrázek 1, kde je znázorněn případ, kdy se nastavuje cílové město (`TO_CITY`) cesty (`TRAVEL`) na `BOSTON`, což je patrné ze symbolu "=", který je označen na horní liště tabulky obsahující tuto podmínku.



Obrázek 1: Ukázka dotazu stanovení podmínky v jazyce QBD [17].

2.3.3 MashQL

MashQL [18] je grafickým dotazovacím jazykem, který pracuje s vizuální reprezentací založenou na diagramech. Tento jazyk funguje na principu překladu dotazů do jazyka SPARQL [19], ve kterém jsou také vykonány.

SPARQL [19] je dotazovací jazyk pro relační datový model patřící k poslednímu doporučení organizace World Wide Web Consortium (W3C) [20], jenž je využíván v případě, kdy jsou data na Internetu uchovávána ve formátu RDF [21] (více k RDF níže). Dotazování probíhá podobně jako v případě jazyka SQL na databáze, avšak principem je dotazování vzdálených RDF zdrojů.

RDF [21] představuje standardní model rovněž vypracovaný organizací W3C, který slouží k výměně dat v prostředí Internetu. RDF se zasloužil o rozšíření propojovací struktury webu tím, že aplikoval URI identifikátory [22], pomocí kterých pojmenovává vztahy mezi věcmi a oběma

konci spojení. Již zmíněná propojovací struktura představuje graf, kde uzly grafu jsou tvořeny jednotlivými zdroji a hrany mezi nimi reprezentují pojmenované spojení.

Na každý dotaz vytvořený dotazovacím jazykem MashQL nahlížíme jako na strom, kde kořen je pojmenován jako předmět dotazu a je jeho základem, dále každá větev tohoto stromu je označována jako omezení dotazu a slouží k omezení přesné vlastnosti předmětu dotazu. Větve stromu mohou samozřejmě také obsahovat podstromy nazývané jako cesty dotazů, které povolují další potřebnou navigaci.

Příklad práce v tomto jazyce můžeme pozorovat na obrázku 2, který představuje uživatelskou reprezentaci pro zadání dotazu na pravé straně a následně vzniklý dotaz převedený do jazyka SPARQL na straně levé. Dotaz spočívá ve vyhledání všech článků autora žijícího na Kypru, které byly publikovány po roce 2000.


```
PREFIX …
SELECT ?ArticleTitle, ?Institute
FROM …
WHERE {
 ?Article :Title ?ArticleTitle
 ?Article :Author ?X1
 ?X1 :Address ?X2
 ?X2 :Country ?X3
 OPTIONAL{?X1 :Affiliation ?Institute}
 ?Article :Year ?X4
 FILTER (?X3 = "Cyprus")
 FILTER (?X4 > 2000)}
```

Obrázek 2: Ukázka dotazu v jazyce MashQL [18].

Na základě výše uvedeného obrázku je patrné, že dotazovací jazyk MashQL poskytuje formulaci dotazů takovým způsobem, aby uživatel nebyl zatěžován složitostí a zároveň také správným pochopením datových struktur, tudíž tuto zodpovědnost přebírá editor dotazů. Jedinou náplní práce uživatelů je tedy vhodné použití položek nazývaných rozbalovací seznamy pro vyjádření potřebného požadavku (dotazu), zatímco editor v souladu s uživatelským výběrem průběžně vykonává různé dotazy na pozadí a dále dynamicky naplňuje daty rozbalovací seznamy.

2.4 Srovnání syntaxe dotazovacích jazyků

Tato část práce je zaměřena na srovnání syntaxe již zmíněných textových dotazovacích jazyků včetně jednoho grafického dotazovacího jazyka QBE s jazykem SQL. U zbylých dvou grafických dotazovacích jazyků QBD a MashQL by byla ukázka syntaxe u jednotlivých dotazů velmi náročná a málo přehledná. Srovnávání je provedeno na datovém modelu zmíněném v kapitole 2.1.

Syntaxe již zmíněných dotazovacích jazyků se od sebe liší, proto je níže poukázáno na odlišnosti mezi nimi při operacích týkajících se čtení záznamu jako jsou projekce a selekce. Tyto

operace byly zvoleny z důvodu zaměření navrhovaných formulářů, které budou sloužit pro dotazování dat z databáze.

2.4.1 Projekce

Vypsání seznamu všech staveb (`typ_stavby`, `ulice`, `cislo_popisne`, `datum_kolaudace`).

SQL:

```
SELECT typ_stavby, ulice, cislo_popisne, datum_kolaudace
FROM Stavba
```

QBE:

K vypsání seznamu všech staveb slouží příkaz "P."(zkratka pro "print") vložený do prvního řádku tabulky a umístěný pod názvem tabulky, viz tabulka 1.

Tabulka 1: Tabulka vypsání záznamu s použitím projekce v jazyce QBE.

Stavba	id_stavby	typ_stavby	ulice	cislo_popisne	datum_kolaudace
		P.	P.	P.	P.

QUEL:

```
RANGE OF s IS Stavba
RETRIEVE (s.typ_stavby, s.ulice, s.cislo_popisne, s.datum_kolaudace)
```

ISBL:

```
LIST Stavba % typ_stavby, ulice, cislo_popisne, datum_kolaudace
```

PIQUE:

```
RETRIEVE (typ_stavby, ulice, cislo_popisne, datum_kolaudace)
```

2.4.2 Selekce

Vypsání seznamu všech dotací (`datum_prideleni`, `způsob_pouziti`), jejichž hodnota je vyšší než 200.000 Kč.

SQL:

```
SELECT datum_prideleni, zpusob_pouziti
FROM Dotace_EU
WHERE vyse_dotace > 200000
```

QBE:

K vypsání pouze některých sloupců je nutné vložit příkaz "P."do těchto sloupců. Uvedené podmínky je nutné vyjádřit u daného sloupce, syntaxe je stejná jako u SQL, viz tabulka 2.

Tabulka 2: Tabulka vypsání záznamu s použitím selekce v jazyce QBE.

Dotace_EU	id_dotace	vyse_dotace	datum_prideleni	zpusob_pouziti	id_stavby
		> 200000	P.	P.	

QUEL:

```
RANGE OF d IS Dotace_EU
RETRIEVE (d.datum_prideleni, d.zpusob_pouziti)
WHERE d.vyse_dotace > 200000
```

ISBL:

```
LIST Dotace_EU : vyse_dotace > 200000 % datum_prideleni, zpusob_pouziti
```

PIQUE:

```
RETRIEVE (datum_prideleni, zpusob_pouziti)
WHERE (vyse_dotace > 200000)
```

2.5 Srovnání dotazovacích jazyků z pohledu přívětivosti k uživatelům

V předchozích kapitolách byly zmíněny významné dotazovací jazyky a následně bylo poukázáno na rozdíly v jejich syntaxích. Dalším důležitým bodem však zůstává, který z těchto dotazovacích jazyků je nejvhodnější a nejvíce přívětivý pro uživatele. Jazyky PIQUE a ISBL byly vyloučeny z důvodu nemožnosti manipulace s daty. Nicméně vyvstala poměrně zajímavá otázka, a to, zda práce s textovými dotazovacími jazyky je pro uživatele jednodušší než práce s vizuálními dotazovacími jazyky. Tato odpověď se vyskytuje ve dvou vypracovaných studiích založených na experimentálním dotazování uživatelů, z nichž první zahrnovala jazyky SQL a QBE [23], kdežto druhá SQL a QBD [24].

Součástí první studie bylo srovnání jazyků SQL a QBE na základě tří hledisek, jednak času potřebného pro formulování dotazu, dále správnosti sestaveného dotazu a v neposlední řadě autorova pohodlí v průběhu utváření dotazu. Testování bylo prováděno nad testovací a kontrolní skupinou, kde každá z nich zahrnovala více než 50 studentů se srovnatelnými vlastnostmi a zkušenostmi v oblasti informatiky. Při porovnání těchto jazyků v rámci vytvoření dotazu v závislosti na čase bylo prokázáno, že vytvoření QBE dotazu s jednoduchou podmínkou v databázi, která je tvořena více tabulkami, je významně rychlejší než jednoduché dotazy v jazyce SQL. Srovnáním jazyků z hlediska správnosti dotazu bylo také zjištěno, že ve větším množství jednotlivých typů dotazů je přesnější řešení v jazyce QBE. Na závěr bylo odvozeno, že spokojenost koncových uživatelů klesá se vzrůstající komplexností dotazů. Při celkovém zhodnocení je tedy možné říci,

že uživatelé jazyka QBE jsou spokojenější při sestavování dotazů a ve většině případech je jejich řešení také přesnější.

Pro přesnější vyhodnocení odlišností mezi textovými a vizuálními dotazovacími jazyky je do této práce zařazeno rovněž porovnání z hlediska jazyků SQL a QBD, které bylo vytvořeno dotazováním uživatelů, s různými znalostmi těchto jazyků, rozdělených do kategorií neznalých, částečně znalých a expertů. U každé skupiny uživatelů bylo měřeno množství chyb při tvorbě dotazu a čas strávený jejich formulací. Bylo však dáno, že u neznalých se bude provádět pouze měření sestavování jednoduchých dotazů, kdežto u lépe znalých uživatelů se budou měřit jak jednoduché, tak složitější dotazy a nakonec u expertů budou zaznamenávány všechny typy dotazů, od nejlehčích po velmi komplexní. V průběhu měření jednoho dotazu v obou jazycích nevznikaly velké rozdíly, jak z pohledu času jeho vytvoření, tak i z hlediska jeho správnosti. Výsledkem tohoto srovnání bylo zjištění, že lepší správnosti dotazu bylo dosaženo v grafickém jazyce QBD, avšak jeho zhotovení bylo zdlouhavější.

2.6 Shrnutí srovnání a výběr dotazovacího jazyka

2.6.1 Rešeršní tabulka

Z předchozích dvou srovnání lze vyvodit tvrzení, že dotazy v grafickém dotazovacím jazyce jsou přesnější a zároveň tito uživatelé jsou mnohem spokojenější s definicí takovýchto dotazů. Proto byl vyvozen závěr, že pro tuto práci bude zvolen grafický dotazovací jazyk. V poslední části rozboru dotazovacích jazyků jsou přehledně porovnány vlastnosti všech zmíněných dotazovacích jazyků (tabulky 3 a 4). Ne všechny zmíněné vlastnosti jsou však stěžejní. Pro účely této práce bylo žádoucí zvolit uživatelsky přívětivý dotazovací jazyk, proto ty vlastnosti, které jsou z hlediska výběru významné, jsou podbarveny.

Tabulka 3: Tabulka srovnání všech zmíněných dotazovacích jazyků. - část 1

Dotazovací jazyky	**Relační DB**	**DDL**	**DML**	**Grafický jazyk**	**Automatická eliminace duplicit**
SQL	✓	✓	✓	x	x
QUEL	✓	✓	✓	x	x
PIQUE	✓	✓	x	x	?
ISBL	✓	✓	x	x	?
QBE	✓	✓	✓	✓	✓
QBD	✓	✓	?	✓	?
MashQL	✓	✓	?	✓	x

Tabulka 4: Tabulka srovnání všech zmíněných dotazovacích jazyků. - část 2

Dotazovací jazyky	**Jednoduchost a intuitivnost**	**Uživatelská dokumentace**	**Laická znalost databáze**	**Vizuální reprezentace bez náhledu tabulek DB**
SQL	✓	✓	x	x
QUEL	✓	✓	x	x
PIQUE	x	x	x	x
ISBL	x	x	x	x
QBE	✓	✓	✓	x
QBD	x	x	x	x
MashQL	✓	✓	✓	✓

2.6.2 Výběr dotazovacího jazyka

Z uvedených rešeršních tabulek 3 a 4 je patrné, že jazyk MashQL má nejlepší hodnocení z grafických dotazovacích jazyků, proto je pro níže uvedené řešení využita obdoba tohoto jazyka. Inspirací byl způsob provedení uživatelského rozhraní, avšak pro samotnou formulaci dotazů není využíván jazyk SPARQL, nýbrž jazyk SQL.

3 Analýza aplikace pro uživatelsky přívětivé dotazování

Náplní této práce bylo vytvoření formulářů pro uživatelsky přívětivé sestavování dotazů nad informačním systémem pro analýzu dat poruch v elektrických sítích využívaným distributory elektrické energie v České i Slovenské republice. Jinými slovy, šlo tedy o navržení formulářů, které uživateli usnadní definici dat vybíraných z databáze, a to takovým způsobem, že uživateli stačí pouze klikat na jednotlivá pole ve formulářích a systém za něj tvoří dotaz pro výběr požadovaných dat z databáze. Uživatel tedy nemusí mít rozsáhlé znalosti z oblasti dotazovacích jazyků, aby byl schopen získat potřebná data z databáze.

3.1 Příklady databází hvězdicového schématu

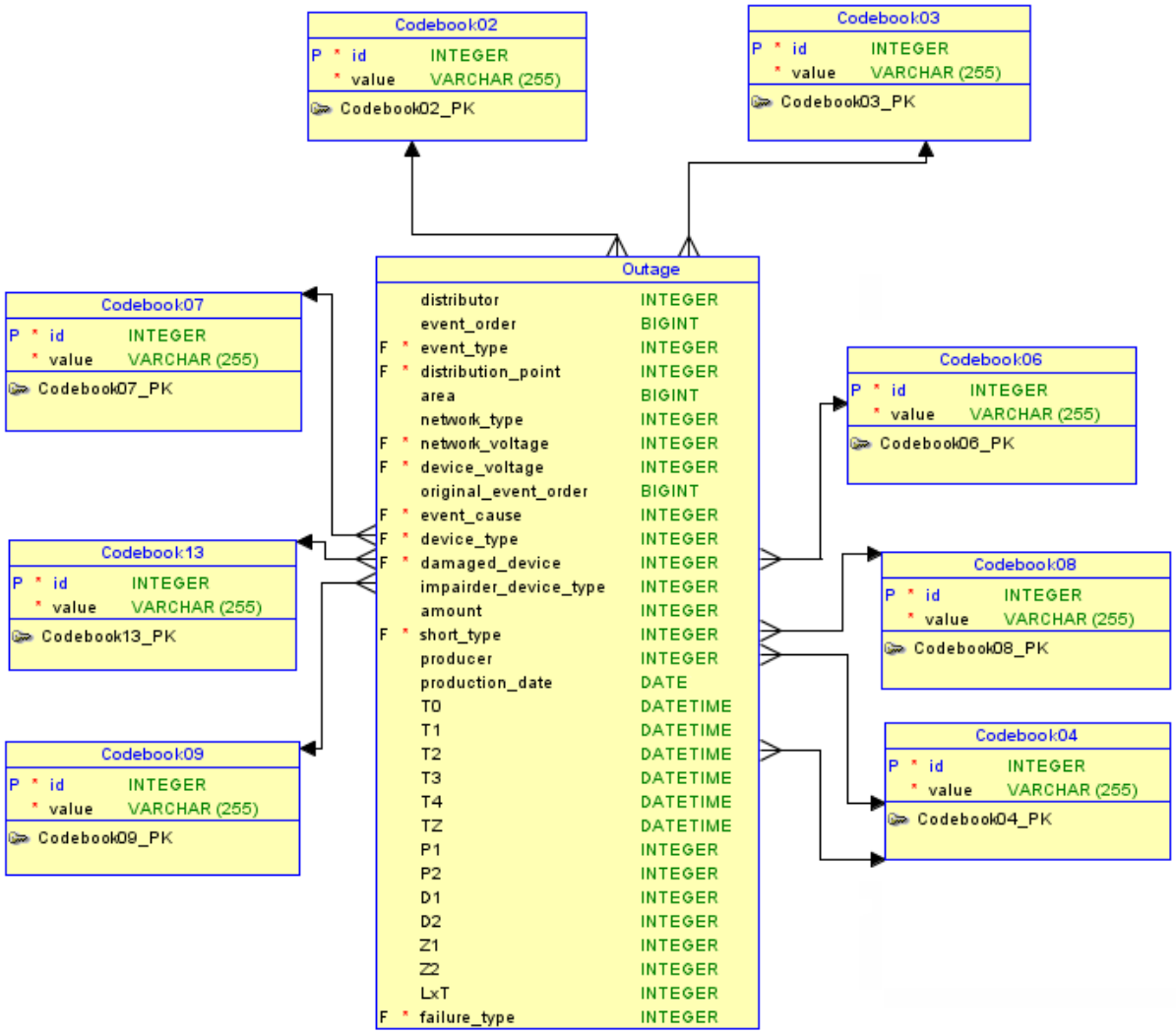
Aplikace byla sice původně navržena pro jeden informační systém, došlo však k jejímu zobecnění v rámci datové vrstvy. Není tedy podstatné, jaká data obsahuje datová vrstva, nýbrž jakého typu je datový model, tedy jak jsou data uspořádaná a uložená v databázi. Navržená aplikace dokáže pracovat se všemi relačními datovými modely, které jsou založeny na hvězdicovém schématu [25]. Toto schéma patří mezi velmi rozšířené typy přístupů při vývoji datových skladů. Základem tohoto schématu je tabulka faktů (jedna nebo více), která odkazuje na různé množství tabulek dimenzí. V tomto případě jsou však tabulky dimenzí reprezentovány vždy jen číselníky. Aplikace byla testována na dvou databázích, které jsou popsány níže. Pro ukládání dat se v obou systémech využívá relační databáze, která je specifikovaná v rámci zadání.

3.1.1 Databáze poruch v elektrických sítích

Tato databáze [26] (obrázek 3) slouží pro sběr dat poruch v elektrických sítích od distributorů elektrické energie v České i Slovenské republice. Základní tabulkou faktů této databáze je relace `Outage`, ke které je vytvořeno velké množství číselníků pro možné dohledání informací na základě identifikační hodnoty. Číselník je často sestavován energetickou regulační kanceláří a tvořen dvojicí hodnot *<identifikační číslo, hodnota>*.

3.1.2 Databáze zákroků veterinární stanice

Výběr této databáze (obrázek 4) byl inspirován jinou bakalářskou prací Univerzity Tomáše Bati ve Zlíně, Fakulty aplikované informatiky, kde student navrhoval databázový systém pro správu veterinární stanice [27]. Z této bakalářské práce je však využita pouze tabulka `Zákroky`, na níž byly provedeny ještě další změny zahrnující mimo jiné také rozšíření tabulky, aby došlo k přizpůsobení schématu databáze navrženému systému, který bylo nutno otestovat použitím jiné databáze. Tabulkou faktů je v tomto případě již zmíněná tabulka `Zákroky`, která obsahuje taktéž odkazy do jiných tabulek - číselníků. Data do této databáze však byla generována náhodně za účelem zajištění anonymity dat.



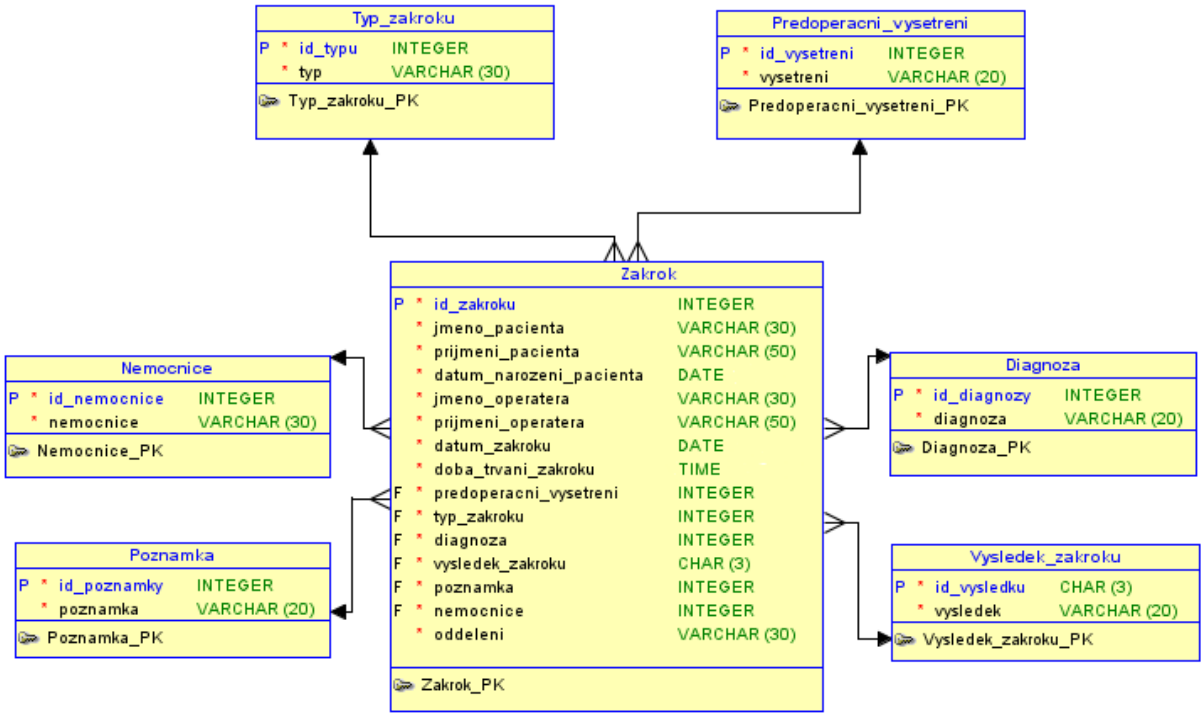
Obrázek 3: Relační model databáze poruch v elektrických sítích [26].

3.2 Požadavky a jejich specifikace

Hlavními důležitými funkcemi pro vyvíjenou aplikaci je výběr požadovaných dat, definice podmínek upravujících hledání a provádění výpočetních či matematických operací nad získávanými daty. Jelikož je aplikace dvojjazyčná (v českém a anglickém jazyce), jsou názvy anglických položek uváděny ve složených závorkách.

3.2.1 Případy užití

V této čísti práce jsou přiblíženy pouze takové funkce systému, které jsou považovány za důležité. Níže jsou popsány případy užití pro výběr požadovaných sloupců pro hledání, stanovení matematické funkce, počitatelné funkce a nakonec také jednoduché podmínky pro sloupec.



Obrázek 4: Relační model databáze zákroků veterinární stanice [27].

Případ užití UC1: Výběr požadovaných sloupců pro hledání

Hlavní aktér: Uživatel
Inicializace: Uživatel zvolí v menu položku “Formulář pro výběr dat z databáze” {“Form for data selection from database”}.
Vstupní podmínky: Žádné.
Základní scénář:

1. Uživatel zahájí výběr stisknutím tlačítka “Přidat projekci” {“Add projection”}.
2. Systém, v případě, kdy dosud nedošlo k žádné úpravě sloupců, zobrazí formulář se všemi zaškrtnutými sloupci hlavní tabulky pro výběr dat, které budou vyhledávány.
3. Uživatel, v případě, že si přeje změnit předvyplnění všech sloupců, odškrtne výběr všech těchto sloupců stisknutím tlačítka “Vymazat výběr všech hodnot” {“Clear projection of all values”}.
   - 3.1 Systém provede odškrtnutí všech zobrazených sloupců.
   - 3.2 Uživatel zvolí požadované sloupce jejich zaškrtnutím.
   - 3.3 Uživatel potvrdí provedené změny stiskem tlačítka “Odeslat filtr” {“Submit filter”}.
   - 3.4 Systém provede kontrolu, zda je zaškrtnut alespoň jeden sloupec.

3.5 Systém, v případě úspěšné kontroly v bodě 3.4, provede aktualizaci vybíraných sloupců v zobrazovaném dotazu do databáze a případ užití končí.

3.6 Systém, v případě neúspěšné kontroly v bodě 3.3, zobrazí informaci "Musí být vybrána hodnota!" {"The value must be selected!"} a scénář pokračuje krokem 3.2.

4. Uživatel, v případě, že si přeje zaškrtnout všechny sloupce, stiskne tlačítko "Vybrat všechny hodnoty" {"Add projection of all values"}.

   4.1 Systém provede zaškrtnutí všech zobrazených sloupců.

   4.2 Uživatel potvrdí provedené změny stiskem tlačítka "Odeslat filtr" {"Submit filter"} a případ užití končí.

5. Uživatel, v případě, že si přeje upravit vybrané sloupce, provede jejich zaškrtnutí/odškrtnutí.

   5.1 Uživatel potvrdí provedené změny stiskem tlačítka "Odeslat filtr" {"Submit filter"}.

   5.2 Systém provede kontrolu, zda je zaškrtnuta alespoň jedna hodnota.

   5.3 Systém, v případě úspěšné kontroly v bodě 5.2, provede aktualizaci vybíraných sloupců a případ užití končí.

   5.4 Systém, v případě neúspěšné kontroly v bodě 5.2, zobrazí informaci "Musí být vybrána hodnota!" {"The value must be selected!"} a scénář pokračuje krokem 5.

6. Uživatel, v případě, že si přeje ponechat nastavení předvyplněných sloupců, stiskne tlačítko "Odeslat filtr" {"Submit filter"}.

Alternativní scénář:

2.1 Systém, v případě, kdy již došlo k úpravě sloupců uživatelem, zobrazí formulář s dříve zvolenými/zaškrtnutými sloupci hlavní tabulky, které uživatel vybral.

2.2 Případ užití pokračuje krokem 3 základního scénáře.

Případ užití UC2: Stanovení matematické funkce pro sloupec

Hlavní aktér: Uživatel
Inicializace: Uživatel zvolí v menu položku "Formulář pro výběr dat z databáze" {"Form for data selection from database"}.
Vstupní podmínky: Žádné.
Základní scénář:

1. Uživatel stiskne tlačítko "Přidat projekci" {"Add projection"} pro volbu práce s vybíranými sloupci.

2. Systém zobrazí formulář s vybranými sloupci hlavní tabulky pro výběr dat.

3. Uživatel zahájí volbu matematické funkce pro sloupec stisknutím tlačítka “Přidat agregační funkci” {“Add aggregation function”}.

4. Systém zobrazí formulář se všemi matematickými funkcemi, které je možné zvolit a použít v rámci jednoho sloupce hlavní tabulky. Mezi nabízené funkce patří “Součet hodnot” {“Sum of values”}, “Průměr hodnot” {“Average of values”}, “Minimum z hodnot” {“Minimum of values”}, “Maximum z hodnot” {“Maximum of values”} a nakonec “Počet záznamů” {“Number of records”}.

5. Uživatel zvolí zaškrtnutím požadovanou funkci.

6. Systém zobrazí rozbalovací seznam, jenž obsahuje veškeré sloupce nabývající pouze numerických hodnot z hlavní tabulky.

7. Uživatel zvolí v rozbalovacím seznamu sloupec, na který chce aplikovat vybranou matematickou funkci.

8. Uživatel potvrdí volbu stisknutím tlačítka “Uložit agregační funkci” {“Save aggregation function”}.

9. Systém provede aktualizaci vybíraných sloupců a zobrazí upravený dotaz do databáze.

10. Uživatel, v případě, že chce zvolit další matematické funkce, pokračuje krokem 5.

Případ užití UC3: Stanovení počitatelné funkce pro sloupec

Hlavní aktér: Uživatel
Inicializace: Uživatel zvolí v menu položku “Formulář pro výběr dat z databáze” {“Form for data selection from database”}.
Vstupní podmínky: Žádné.
Základní scénář:

1. Uživatel stiskne tlačítko “Přidat projekci” {“Add projection”} pro volbu práce s vybíranými sloupci.

2. Systém zobrazí formulář s vybranými sloupci hlavní tabulky pro výběr dat.

3. Uživatel zahájí volbu počitatelné funkce pro možnost práce se dvěma sloupci stisknutím tlačítka “Přidat počitatelnou funcki” {“Add computable function”}.

4. Systém zobrazí formulář s možností volby různých typů hodnot, které chce uživatel zpracovávat. Jsou zobrazeny volby “Hodnoty vyjádřené ve tvaru datum a čas” {“Date and time values”} a “Hodnoty numerické” {“Numeric values”}.

5. Uživatel zvolí požadovanou volbu, se kterou chce dále pracovat.

6. Systém zobrazí formulář pro počitatelné funkce obsahující 5 rozbalovacích seznamů a jedno textové pole (zvolena volba “Hodnoty vyjádřené ve tvaru datum a čas” {“Date and time values”}) nebo 4 rozbalovací seznamy a jedno textové pole (zvolena volba “Hodnoty numerické” {“Numeric values”}). Zmíněné položky jsou zobrazeny v následujícím pořadí (zleva doprava):

    - první rozbalovací seznam - stanovení prvního sloupce
    - druhý rozbalovací seznam - stanovení operace pro vybrané sloupce
    - třetí rozbalovací seznam - stanovení druhého sloupce
    - čtvrtý rozbalovací seznam - stanovení operace porovnání
    - textové pole - stanovení porovnávané hodnoty
    - pátý rozbalovací seznam - stanovení porovnávané jednotky (týká se pouze volby “Hodnoty vyjádřené ve tvaru datum a čas” {“Date and time values”}): “rok” {“year”}, “měsíc” {“month”}, “den” {“day”}, “hodina” {“hour”}, “minuta” {“minute”} a “sekunda” {“second”}

7. Uživatel zadá požadovanou počitatelnou funkci.

8. Uživatel uloží vytvořenou funkci stisknutím tlačítka “Uložit počitatelnou funkci” {“Save computable function”}.

9. Systém provede aktualizaci zobrazovaného dotazu do databáze přidáním zvolené počitatelné funkce.

10. Uživatel, v případě, že chce zvolit další matematické funkce, pokračuje krokem 5.

Případ užití UC4: Stanovení jednoduché podmínky pro sloupec

Hlavní aktér: Uživatel
Inicializace: Uživatel zvolí v menu položku “Formulář pro výběr dat z databáze” {“Form for data selection from database”}.
Vstupní podmínky: Žádné.
Základní scénář:

1. Uživatel zahájí výběr sloupce, popřípadě sloupců, nad kterým chce definovat podmínku, stisknutím tlačítka “Přidat filtr” {“Add filter”}.

2. Systém, v případě, kdy dosud nedošlo k žádnému výběru sloupců, zobrazí formulář se zaškrtávacími tlačítky reprezentující všechny sloupce hlavní tabulky. Všechna tato tlačítka jsou nezaškrtnutá.

3. Uživatel zvolí sloupec nebo sloupce, pro které chce stanovit podmínku, zaškrtnutím příslušného tlačítka s názvem daného sloupce.

4. Uživatel potvrdí volbu stiskem tlačítka “Odeslat filtr” {“Submit filter”}.

5. Systém zobrazí formulář se všemi uživatelem vybranými sloupci v řádcích pod sebou, kde u každého sloupce vykreslí tlačítko “Zvolit” {“Choose”}.

6. Uživatel stiskne tlačítko “Zvolit” {“Choose”} u požadovaného sloupce, kterému chce přiřadit či upravit podmínku.

7. Systém zobrazí dvě zaškrtávací tlačítka “Jednoduchá podmínka” {“Simple condition”} a “Komplexní podmínka” {“Complex condition”}.

8. Uživatel provede volbu jednoduché podmínky stiskem tlačítka “Jednoduchá podmínka” {“Simple condition”}.

9. Systém, v případě, kdy se jedná o sloupec, který nabývá:

   9.1 celočíselných hodnot, zobrazí pole s možností zadat pouze celočíselnou hodnotu.

   9.2 desetinných hodnot, zobrazí pole s možností zadat desetinné číslo (desetinná čárka je prezentována tečkou).

   9.3 textových hodnot, zobrazí pole pro zadání textu pomocí jakýchkoliv znaků.

   9.4 hodnot ve tvaru datum i čas, zobrazí dvě pole. První pole s rozbalovacím kalendářem slouží pro zadání data ve formátu “rrrr-mm-dd” {“yyyy-mm-dd”} a druhé pole pro zadání času ve 24 hodinovém formátu “hh:mm:ss”. Je možné vyplnit pouze jedno z polí.

   9.5 hodnot pouze ve tvaru datum, zobrazí pole s rozbalovacím kalendářem pro zadání data ve formátu “rrrr-mm-dd” {“yyyy-mm-dd”}.

   9.6 hodnot pouze ve tvaru času, zobrazí pole pro zadání času ve 24 hodinovém formátu “hh:mm:ss”.

   9.7 pravdivostních hodnot, zobrazí rozbalovací seznam se dvěma možnostmi “pravda” {“true”} a “nepravda” {“false”}.

   9.8 pevně určených hodnot, zobrazí rozbalovací seznam s obsahem všech těchto daných možností.

10. Uživatel zadá požadovanou hodnotu.

11. Uživatel potvrdí zadanou hodnotu stiskem tlačítka “Potvrdit jednoduchou podmínku” {“Confirm simple condition”}.

12. Systém provede kontrolu na správně zadanou hodnotu v poli/polích.

13. Systém, v případě úspěšné kontroly popsané v bodě 14, provede vytvoření podmínky pro zvolený sloupec.

14. Systém zobrazí sloupce, které uživatel vybral pro stanovení podmínky. Vedle upravovaného sloupce se zobrazí uživatelem vytvořená podmínka.

15. Systém provede aktualizaci zobrazovaného dotazu do databáze přidáním utvořené jednoduché podmínky.

Alternativní scénář:

2.1 Systém, v případě, kdy již byl proveden výběr některých sloupců, zobrazí formulář s dříve zvolenými/zaškrtnutými sloupci hlavní tabulky, které uživatel vybral.

3.1 Uživatel, v případě, že chce odstranit některý z vybraných sloupců, odškrtne příslušné tlačítko s názvem daného sloupce.

12.1 Systém, v případě neúspěšné kontroly v bodě 12, zobrazí chybovou zprávu o nesprávné hodnotě a neumožní uložení podmínky. Uživatel musí opravit zadanou hodnotu a poté případ užití pokračuje krokem 11 základního scénáře.

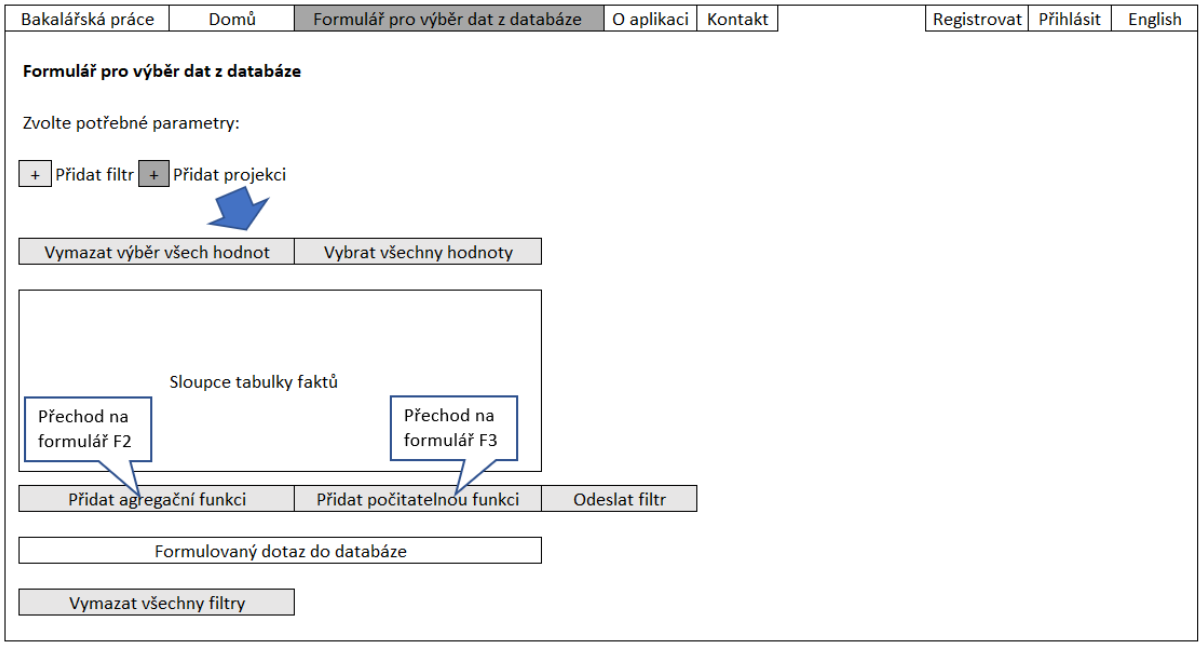
4 Návrh aplikace pro uživatelsky přívětivé dotazování

4.1 Vstupní a výstupní data aplikace

Při spuštění aplikace dochází k analýze vstupních parametrů, z důvodu správného navržení uživatelského rozhraní pro danou databázi. Mezi vstupní parametry patří tedy skript pro vytvoření databáze, nad kterou je daná aplikace spouštěna, a samozřejmě je nutností, aby databáze obsahovala data, jinak z navrženého formuláře nebude možné je vyhledávat či se na ně dotazovat. Ze skriptu pro vytvoření databáze aplikace zjistí názvy sloupců hlavní tabulky (tabulky faktů) a jejich hodnoty pro správné stanovení filtru nebo projekce a výběr vhodného formuláře k volbě podmínky každého sloupce. Výstupem jsou načtená data vrácená na základě uživatelsky definovaného výběru dat.

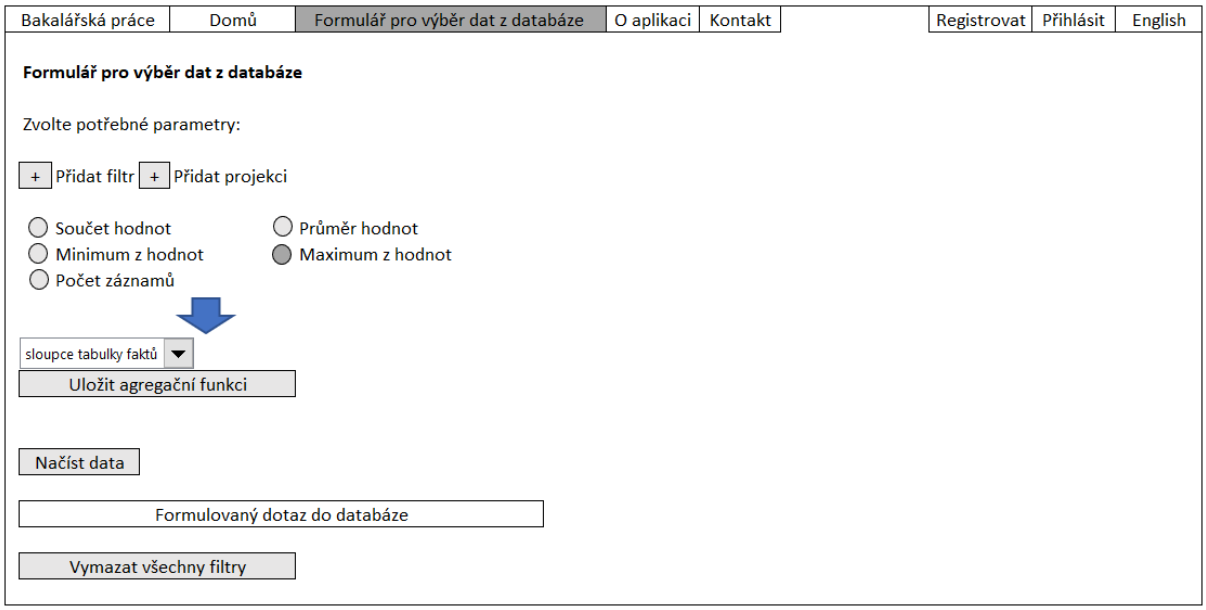
4.2 Návrh uživatelského rozhraní

Dalším nutným krokem při vývoji aplikace je samozřejmě také návrh uživatelského rozhraní [28]. V rámci této bakalářské práce budou formuláře uvedené níže prezentovány pouze v českém jazyce.



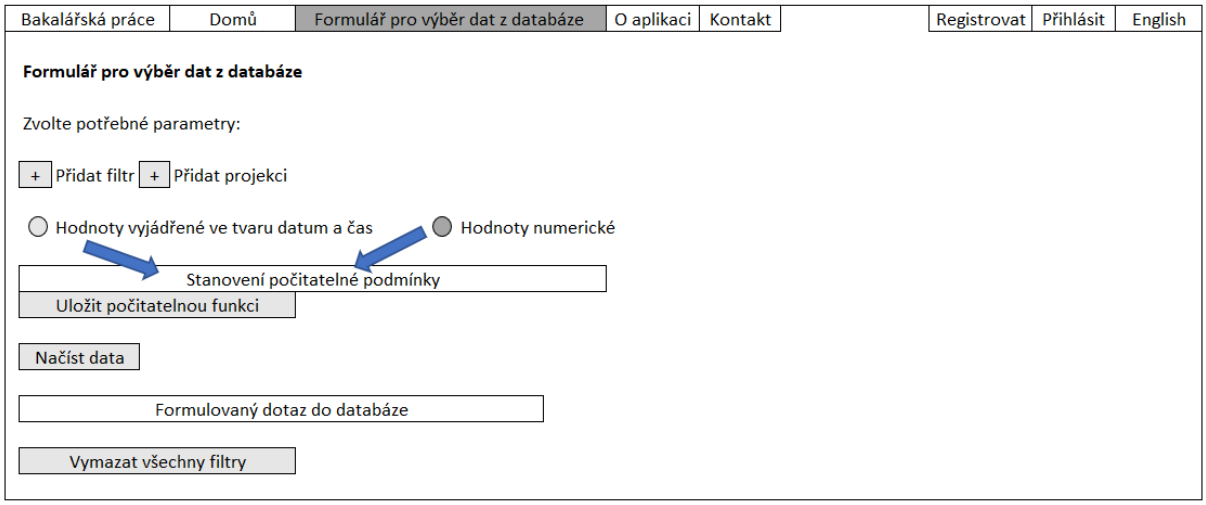
Obrázek 5: Formulář F1 k výběru požadovaných sloupců pro hledání.

Na obrázku 5 je zobrazen formulář F1 k výběru požadovaných sloupců pro hledání. Práce s daným formulářem je popsána v rámci případu užití UC1 v kapitole 3.2.1. Zároveň tento formulář slouží jako vstupní bod pro následující dva formuláře F2 a F3.



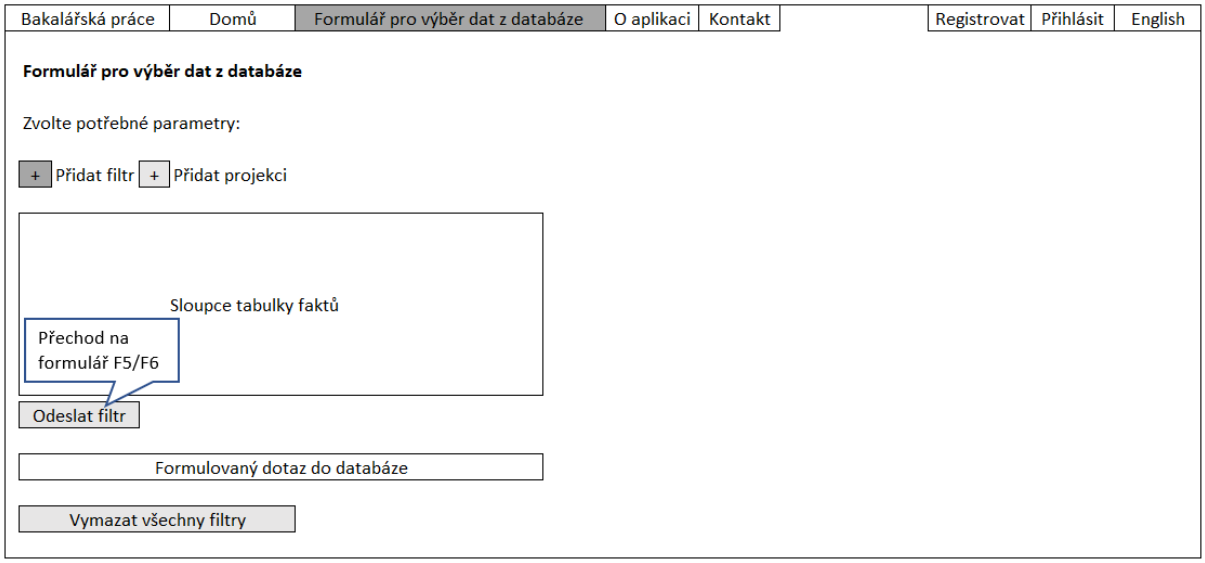
Obrázek 6: Formulář F2 ke stanovení matematické funkce pro sloupec.

Obrázek 6 vykresluje formulář F2 ke stanovení matematické funkce pro sloupec. Funkčnost, kterou daný formulář zajišťuje je popsána v rámci případu užití UC2 v kapitole 3.2.1.

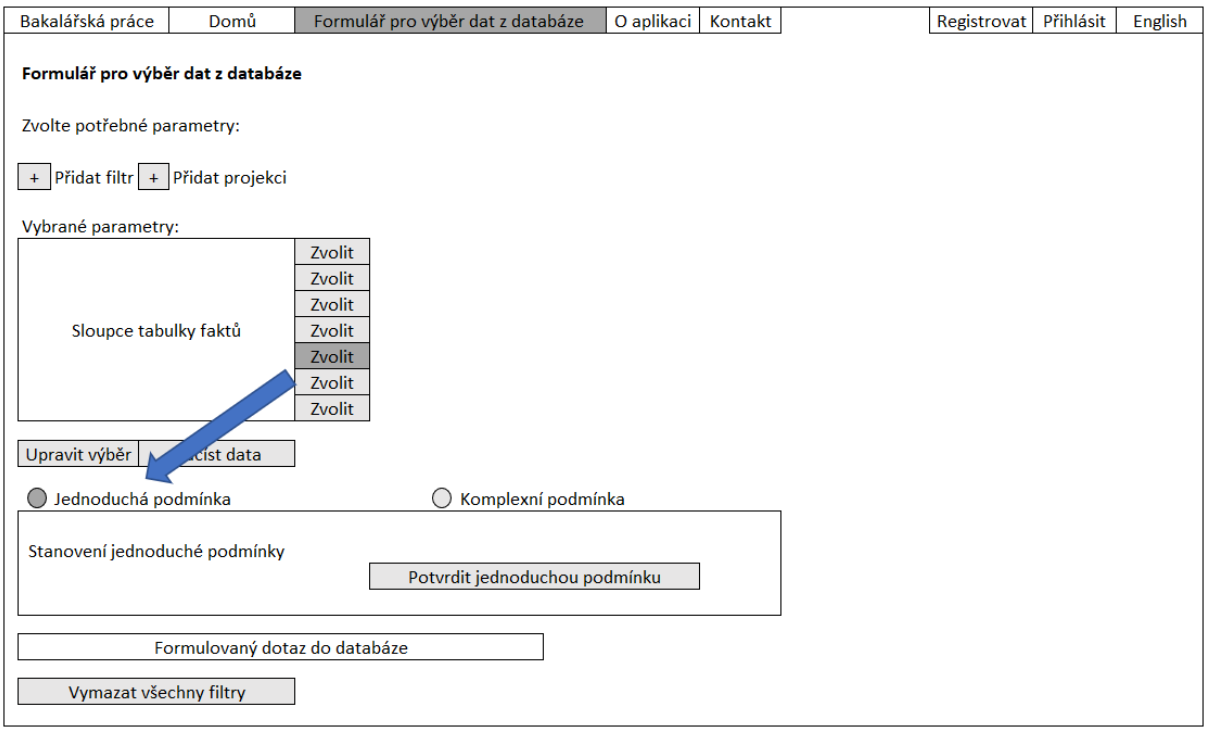


Obrázek 7: Formulář F3 ke stanovení počitatelné funkce pro sloupec.

Na obrázku 7 je zobrazen formulář F3 ke stanovení počitatelné funkce pro sloupec. Funkčnost poskytovaná tímto formulářem je popsána v rámci případu užití UC3 v kapitole 3.2.1.

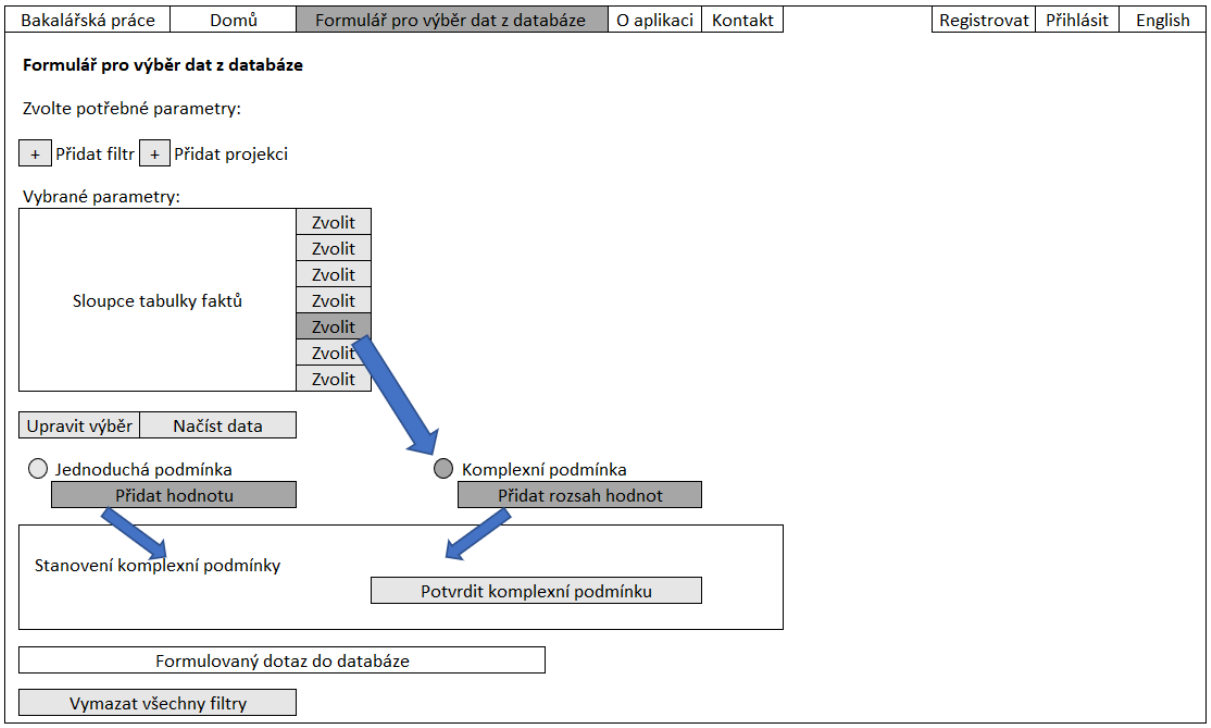


Obrázek 8: Formulář F4 k výběru požadovaných sloupců pro stanovení podmínky.



Obrázek 9: Formulář F5 ke stanovení jednoduché podmínky pro sloupec.

Poslední tři obrázky vykreslují formulář F4 k výběru požadovaných sloupců pro stanovení podmínky (obrázek 8), formulář F5 ke stanovení jednoduché podmínky pro sloupec (obrázek 9) a formulář F6 ke stanovení komplexní podmínky pro sloupec (obrázek 10). Formulář F4 představuje vstupní bod k dalším dvěma formulářům F5 a F6. Funkčnost zajišťovaná spojením formulářů F4 a F5 je popsána v rámci případu užití UC4 v kapitole 3.2.1. Poslední formulář F6 umožňuje



Obrázek 10: Formulář F6 ke stanovení komplexní podmínky pro sloupec.

stanovení komplexní podmínky postupným klikáním na tlačítka “Přidat hodnotu” a “Přidat rozsah hodnot”, na základě kterých jsou generována požadovaná pole. Zobrazení těchto polí je řízeno stejnými podmínkami jako v rámci případu užití UC4 (tzn. v závislosti na hodnotách obsažených ve sloupci, pro který je podmínka tvořena).

5 Implementace aplikace pro uživatelsky přívětivé dotazování

5.1 Použité technologie

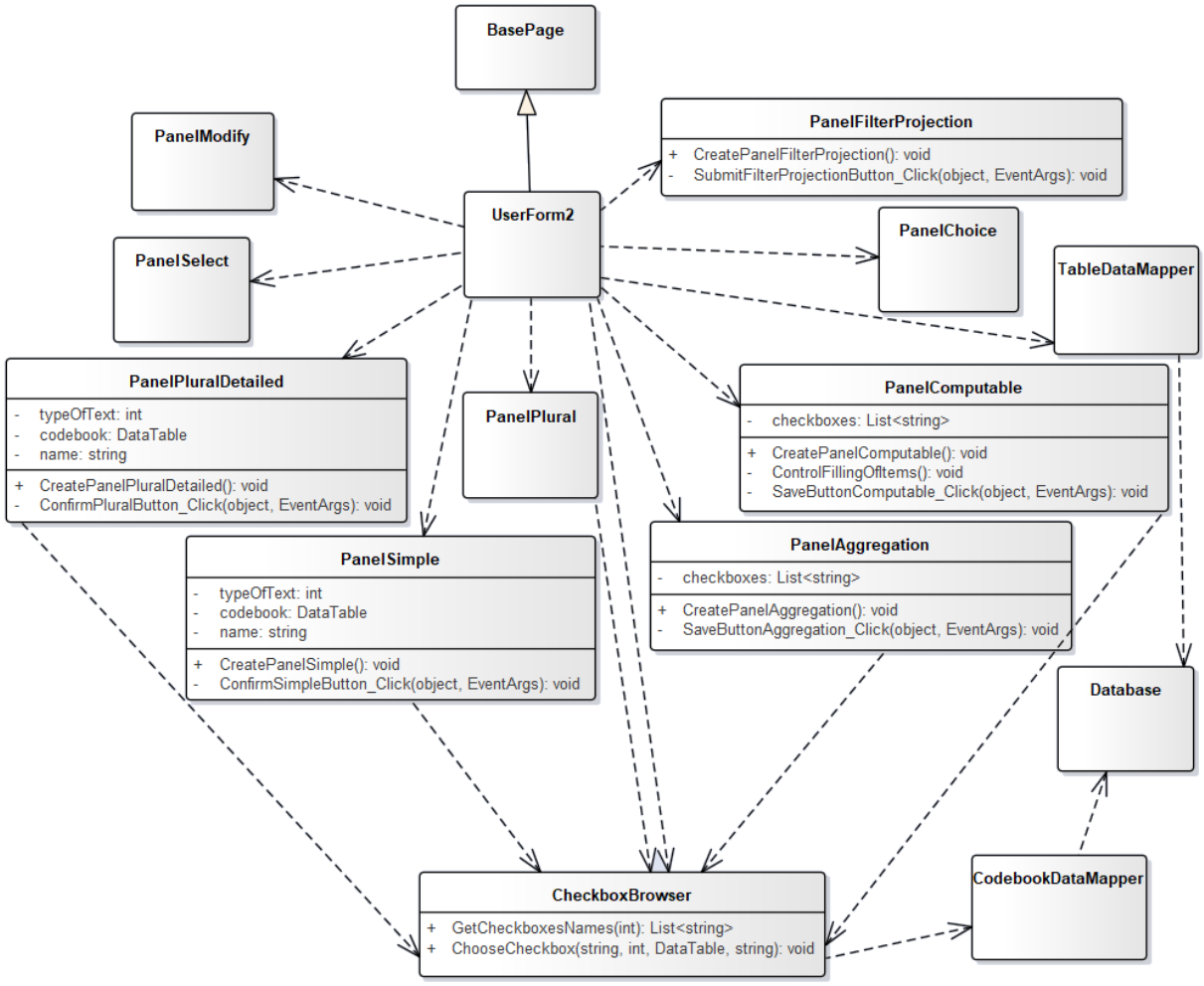
Za účelem snazší integrace navrženého formuláře do již existujícího řešení IS, který je vytvořen jako webová aplikace ve vývojovém prostředí Visual Studio, bylo tedy potřeba zvolit stejnou technologii pro vývoj. Celá aplikace je proto navržena v jazyce c#, konkrétně v technologii ASP.NET [29] nazývané WebForms [30]. Základem technologie WebForms je protokol HTTP [31]. Dotaz do databáze, který je vytvořen v rámci běhu aplikace je definován v jazyce SQL nad SQL Serverem od firmy Microsoft [32].

HTTP je internetový protokol běžící na portu TCP/80, založený na komunikaci mezi klientem a serverem formou požadavek-odpověď. Můžeme však narazit na problém, pokud budeme chtít uchovávat hodnoty proměnných během interakce s uživatelem, neboť tento protokol je bezestavový, což znamená, že každým dalším požadavkem dochází ke ztrátám předchozích informací uložených na serveru. Proto tento protokol umožňuje použít tzv. session za účelem uchování důležitých informací o uživateli.

WebForms [30] je technologie založená na webových formulářích (Forms). Existují dva typy těchto formulářů - *Web Form* a *Web Form with Master Page*. První typ představuje klasický formulář, jehož obsah si nadefinujeme, kdežto druhý typ má kromě námi nadefinovaného obsahu také stanoven předpis (definován ve třídě `Site.Master`), který bude zobrazen na všech daných formulářích. V rámci této práce byla však zvolena implementace s použitím *Web Form with Master Page*, aby bylo menu pro orientaci přístupné ze všech formulářů v aplikaci. Jelikož technologie WebForms vychází z protokolu HTTP, umožňuje také pracovat s tzv. `Session`, do které je možné vkládat informace podstatné pro běh aplikace, aby nedošlo k jejich ztrátám během interakce s uživatelem. V závislosti na zaměření této aplikace, kdy je kladen důraz na uchování hodnot zadaných uživatelem, je často využívána konstrukce `Session`. Pro nastavení parametrů a spuštění samotné webové aplikace jsou důležité třídy `Startup` a `Global.asax`. Třída `Global.asax` obsahuje podstatné metody `Application_Start` (spuštění aplikace) a `Session_Start` (spuštění `Session`).

5.2 Třídní diagram

Třídní diagram na obrázku 11 zobrazuje pouze metody a proměnné, které jsou níže zmíněny v bližším popisu některých tříd. Ze zmíněného diagramu je také patrné, že aplikace je rozdělen do několika tříd, přičemž největší pozornost patří třídě, reprezentující webový formulář, `UserForm2` sloužící pro vykreslování jednotlivých panelů, formování dotazu do databáze a nakonec prezentování požadovaných dat na stránce. Tato třída tedy tvoří základ celé aplikace. Jelikož je vytvořená aplikace dvojjazyčná, vzniká potřeba ošetření přepínání jazyků, z toho důvodu třída `UserForm2` dědí ze třídy `BasePage`, jež zajišťuje nastavení aktuálně používaného jazyka.



Obrázek 11: Třídní diagram zhotoveného systému.

Při spouštění aplikace je kladen důraz na třídu `Reader` (není zmíněna v třídním diagramu, neboť je volána ve třídě `Global.asax`, v metodě `Application_Start`), jejímž úkolem je zpracovat načtený soubor ze složky `App_Data` [33], která představuje úložiště datových souborů aplikace. Z tohoto souboru, obsahujícího skript pro vytvoření tabulky faktů databáze, třída vyčlení důležité informace ohledně dotazované databáze jako jsou názvy sloupců včetně jejich datových typů, popř. názvy tabulek dimenzí odkazovaných zmíněnými sloupci.

Třída `CheckboxBrowser`, jedna z nejzajímavějších tříd z hlediska běhu aplikace, prochází sloupce načtené při spuštění aplikace a poskytuje informace o nich jiným třídám, které je využívají. Mezi další velice významné třídy spadají `PanelFilterProjection`, `PanelAggregation` a `PanelComputable` nabízející uživateli možnost vybrat sloupce, s nimiž chce dále pracovat a definovat nad nimi počitatelné či matematické funkce.

Úkolem třídy `PanelSelect` je poskytování sloupců, vybraných pro volbu podmínky, zobrazením:

- zaškrtávacích tlačítek ze `Session["FilterCheckboxes"]` poskytnutých třídami `PanelFilterProjection` nebo `PanelComputable`
- popisu definované podmínky každého sloupce ze `Session["SelectLabels"]`
- tlačítka umožňujícího definici/změnu podmínky

Mezi třídy, podílející se na samotné definici podmínky, patří `PanelChoice`, která slouží jen pro stanovení typu podmínky (jednoduchá či komplexní), `PanelSimple`, jež vytváří jednoduchou podmínku, a nakonec `PanelPlural` spolu s `PanelPluralDetailed` zajišťující tvorbu komplexní podmínky. `PanelPlural` zajišťuje opět jednoduchou funkčnost, a to zobrazení výběru při formulaci podmínky, ze kterého je možné zvolit přidání rozsahu hodnot (např. hodnota v rozmezí 1 a 2) či přidání pouze jedné hodnoty. Na základě tohoto výběru dochází k vložení hodnoty 1 (hodnota) nebo 2 (rozsah hodnot) do `Session["Values"]`.

Jednou z posledních tříd je `PanelModify`, která umožňuje uživateli upravit logické operátory (a, nebo), spojující jednotlivé podmínky, u již vytvořeného dotazu. Nakonec samozřejmě nesmí chybět třídy umožňující načítání sloupců tabulky faktů a tabulek dimenzí z databáze, jimiž jsou `TableDataMapper` a `CodebookDataMapper`. Tyto třídy využívají funkčnosti třídy `Database`, jež zajišťuje připojení k databázi a komunikaci s ní. V dalších částech této kapitoly budou popsány rozsáhlé třídy důležité pro funkčnost aplikace, které byly zatím jen lehce zmíněny.

5.2.1 Třída PanelFilterProjection

Zaštiťuje funkčnost dvou panelů, a to jak panelu pro výběr sloupců k hledání (tzv. projekci), tak panelu pro definici sloupců, kterým bude přiřazena podmínka (tzv. selekce). Metoda `CreatePanelFilterProjection`, v případě obou panelů, zajišťuje zobrazení zaškrtávacích tlačítek s názvy sloupců tabulky faktů a potvrzovacího tlačítka. Při použití panelu pro projekci rovněž vykresluje tlačítka k výběru počitatelné i matematické funkce. Metoda `SubmitFilterProjectionButton_Click`, inicializovaná kliknutím na potvrzovací tlačítko, nahrává položky zaškrtnuté uživatelem do `Session["FilterCheckboxes"]` nebo `Session["ProjectionCheckboxes"]` v závislosti na panelu, který je aktuálně využíván, aby při dalším načtení této stránky mohly být předem označeny již vybrané sloupce z předchozích interakcí.

5.2.2 Třída PanelAggregation

Již v samotném konstruktoru této třídy dochází k načtení všech numerických sloupců tabulky faktů (tzn. volání statické metody `GetCheckboxesNames()` s parametrem 1 třídy `CheckboxBrowser`), aby je bylo možné zobrazit v rozbalovacím seznamu při definici

funkce. Zobrazovací metoda `CreatePanelAggregation` zajišťuje vykreslení zaškrtávacích tlačítek všech dostupných agregačních funkcí (tj. SUM, AVG, MIN, MAX, COUNT), popřípadě, pokud byl již proveden výběr funkce zaškrtnutím příslušného tlačítka, vykresluje také sloupce tabulky faktů v rozbalovacím seznamu včetně potvrzovacího tlačítka. Metoda `SaveButtonAggregation_Click`, která je spouštěna po stisku potvrzovacího tlačítka, provádí vložení zformulované části dotazu týkající se vybrané funkce do `Session["AggregationFunction"]` mající formu řetězce, kde se přidávají všechny agregační funkce zadané uživatelem po dobu běhu aplikace.

5.2.3 Třída PanelComputable

V konstruktoru této třídy dochází k načtení buď numerických sloupců (jako u třídy `PanelAggregation`), anebo sloupců obsahujících položky s datem (voláním metody `GetCheckboxesNames()` třídy `CheckboxBrowser`, ale s parametrem 2) na základě toho, s jakými sloupci se uživatel rozhodl pracovat. Pro zobrazení položek je využita metoda `CreatePanelComputable`, která vykresluje, jak zaškrtávací tlačítka pro volbu typu sloupců (numerických/ve tvaru datum a čas), se kterými se uživatel chystá pracovat, tak v případě jeho rozhodnutí také rozbalovací seznamy obsahující:

1. sloupce načtené v konstruktoru,
2. operace pro práci s těmito sloupci,
3. porovnávací operátory,
4. porovnávané jednotky (rok, měsíc, den, hodina, minuta, sekunda) - pouze u sloupců s datem nebo časem.

Mimo zmíněné rozbalovací seznamy je také zobrazeno textové pole k zadání porovnávané hodnoty a potvrzovací tlačítko. V případě, že dochází k úpravě již nastavené počitatelné funkce vykreslené třídou `PanelSelect`, dochází ještě k volání metody `ControlFillingOfItems()`, která předvyplní vyobrazené položky pro usnadnění práce uživateli. Potvrzovací metoda `SaveButtonComputable_Click`, reagující na stisknutí tlačítka potvrzení výběru, vkládá část dotazu utvořenou pro projekci do `Session["ComputableFunction"]`, kde jsou vkládány informace jako samostatné řetězce, kdežto další část dotazu, obsahující podmínku, je vložena do samostatných Session. Nakonec ještě vytváří zaškrtávací tlačítko, reprezentující zadané dva sloupce s definovanou operací mezi nimi, které přidává do `Session["FilterCheckboxes"]`, se kterou je pracováno také ve třídě `PanelFilterProjection` (popsané výše) a kde jsou vkládány jednotlivé sloupce pro definici podmínky. Mimo zmíněné zaškrtávací tlačítko metoda ještě generuje popisné pole s uživatelským popisem vytvořené funkce, které přidává do `Session["SelectLabels"]`.

5.2.4 Třída PanelSimple a PanelPluralDetailed

Hned v konstruktoru probíhá načtení podstatných informací o vybraném sloupci, pro který se stanovuje podmínka, voláním metody `ChooseCheckbox()` (třídy `CheckboxBrowser`) se čtyřmi parametry, kde první parametr je identifikační hodnota vybraného sloupce, zbylé tři parametry (datový typ sloupce = `typeOfText`; výpis všech sloupců tabulky dimenzí, pokud pro daný sloupec existuje = `codebook`; název sloupce = `name`) jsou proměnnými této třídy, které jsou naplněny volanou metodou a vráceny zpět. V metodách `CreatePanelSimple` a `CreatePanelPluralDetailed` probíhá zobrazování polí k definici podmínky s ohledem na hodnoty proměnných získané v konstruktoru, ze kterých má největší význam proměnná `typeOfText`, neboť dle její hodnoty dochází ke generování:

- rozbalovacích seznamů (pokud vybraný sloupec odkazuje do tabulky dimenzí),
- textových polí s validacemi:
  - celočíselných hodnot,
  - desetinných hodnot s desetinnou tečkou,
  - časových hodnot ve formátu "hh:mm:ss",
- textových polí s rozbalovacím kalendářem a validací formátu "yyyy-mm-dd".

V případě metody `CreatePanelPluralDetailed` je uskutečňováno ještě navíc rozhodnutí, která pole generovat, jestli pole pro zadání jedné hodnoty anebo rozsahu hodnot. Tento proces spočívá v načtení hodnot z `Session["Values"]` (naplněné ve třídě `PanelPlural`). Metody `ConfirmSimpleButton_Click` a `ConfirmPluralButton_Click` ukládají zformované podmínky dotazu do samostatných Session pro každý sloupec a definují uživatelský popis podmínky, který je pak přiřazen do `Session["SelectLabels"]`.

6 Ukázka práce s aplikací

Pro názornou ukázku práce s vyvinutou aplikací jsou zde vloženy obrázky, které prezentují funkčnost samotné aplikace na několika scénářích v rámci odlišných databází. Výsledkem každého scénáře je vyhledání dat z používané databáze na základě stanovených podmínek či projekce. Tato data jsou zobrazena v tabulce, kde sloupce, nad kterými je definovaná podmínka, jsou podbarveny a ve svém záhlaví obsahují specifikaci této podmínky. Samotný dotaz sestavený na základě klikání uživatele je umístěný nad zmíněnou tabulkou výpisu dat a je možné v něm před samotným načtením dat upravit logické operátory mezi jednotlivými podmínkami specifikovanými uživatelem. Pod tímto dotazem jsou také umístěna data popisující počet nalezených záznamů, dobu zpracování dotazu a chyby (pokud se vyskytnou) vzniklé v případě zpracování dotazu na úrovni databáze. Z důvodu velkého množství sloupců v tabulkách faktů každé databáze je v níže uvedených formulářích zavedena projekce (dle kroků označených ve formuláři F1 v kapitole 4.2).

Formulář pro výběr dat z databáze

Zvolte potřebné parametry:

\+ Přidat filtr + Přidat projekci

Vybrané parametry:
event order = 4096501939 ∨ (4096519342<=event order<=4096561304) Zvolit
event type = plánovaná Zvolit
device voltage = 22 Zvolit
failure type = E1 - poruchy bez poškození zařízení Zvolit

Upravit výběr Načíst data

SELECT distributor, event_order, event_type, device_voltage, device_type, failure_type FROM Outage WHERE(((event_order=4096501939)OR(event_order BETWEEN 4096519342 AND 4096561304)))AND((event_type='2'))AND((device_voltage='5'))OR((failure_type='1'))
Počet záznamů: 47555
Doba zpracování dotazu: 0 minut 30 sekund

Vymazat všechny filtry

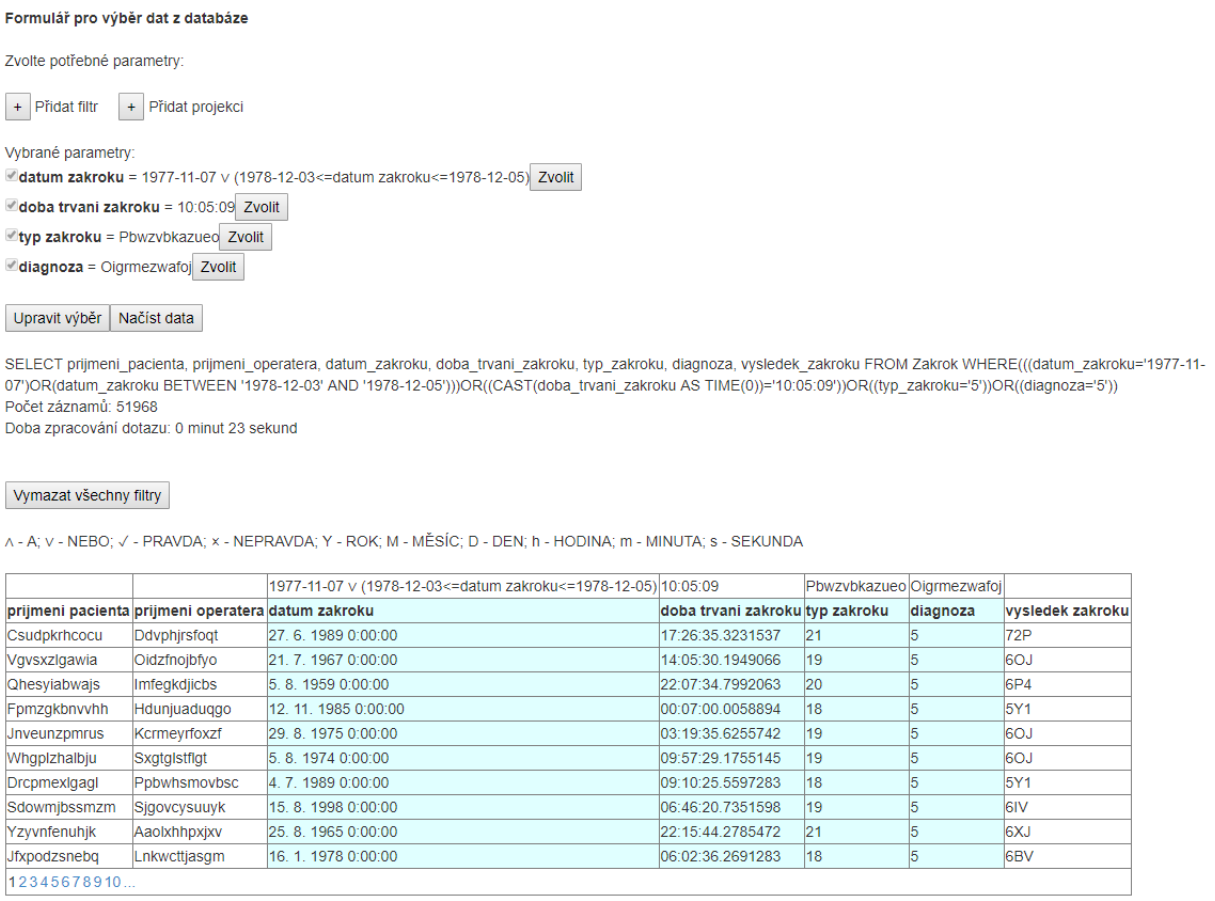
∧ - A; ∨ - NEBO; ✓ - PRAVDA; × - NEPRAVDA; Y - ROK; M - MĚSÍC; D - DEN; h - HODINA; m - MINUTA; s - SEKUNDA

	4096501939 ∨ (4096519342<=event order<=4096561304)	plánovaná	22		E1 - poruchy bez poškození zařízení
distributor	**event order**	**event type**	**device voltage**	**device type**	**failure type**
5	4096501939	1	5	1	1
5	4096497186	1	5	1	1
5	4096492569	1	5	1	1
5	4096500525	1	5	7	1
5	4096507424	1	5	1	1
5	4096456966	1	5	7	1
5	4096468990	1	5	7	1
5	4096484814	1	5	7	1
5	4096498374	1	5	5	1
5	4096391828	1	7	7	1
1 2 3 4 5 6 7 8 9 10 ...					

Obrázek 12: Náhled scénáře 1 pro vyhledání dat z DB poruch v elektrických sítích dle definovaných podmínek.

První scénář (obrázek 12) zobrazuje vyhledání dat z databáze poruch v elektrických sítích na základě definovaných jednoduchých a komplexních podmínek, kdy postupné kroky vedoucí

k vytvoření daného dotazu jsou naznačeny na formulářích F4 až F6 v kapitole 4.2. Stejný typ scénáře pro databázi zákroků veterinární stanice je vykreslen na obrázku 13.



Formulář pro výběr dat z databáze

Zvolte potřebné parametry:

+ Přidat filtr + Přidat projekci

Vybrané parametry:

datum zakroku = 1977-11-07 ∨ (1978-12-03<=datum zakroku<=1978-12-05) Zvolit

doba trvani zakroku = 10:05:09 Zvolit

typ zakroku = Pbwzvbkazueo Zvolit

diagnoza = Oigrmezwafoj Zvolit

Upravit výběr Načíst data

SELECT prijmeni_pacienta, prijmeni_operatera, datum_zakroku, doba_trvani_zakroku, typ_zakroku, diagnoza, vysledek_zakroku FROM Zakrok WHERE(((datum_zakroku='1977-11-07')OR(datum_zakroku BETWEEN '1978-12-03' AND '1978-12-05')))OR((CAST(doba_trvani_zakroku AS TIME(0))='10:05:09'))OR((typ_zakroku='5'))OR((diagnoza='5'))
Počet záznamů: 51968
Doba zpracování dotazu: 0 minut 23 sekund

Vymazat všechny filtry

∧ - A; ∨ - NEBO; ✓ - PRAVDA; × - NEPRAVDA; Y - ROK; M - MĚSÍC; D - DEN; h - HODINA; m - MINUTA; s - SEKUNDA

		1977-11-07 ∨ (1978-12-03<=datum zakroku<=1978-12-05)	10:05:09	Pbwzvbkazueo	Oigrmezwafoj	
prijmeni pacienta	**prijmeni operatera**	**datum zakroku**	**doba trvani zakroku**	**typ zakroku**	**diagnoza**	**vysledek zakroku**
Csudpkrhcocu	Ddvphjrsfoqt	27. 6. 1989 0:00:00	17:26:35.3231537	21	5	72P
Vgvsxzlgawia	Oidzfnojbfyo	21. 7. 1967 0:00:00	14:05:30.1949066	19	5	6OJ
Qhesyiabwajs	Imfegkdjicbs	5. 8. 1959 0:00:00	22:07:34.7992063	20	5	6P4
Fpmzgkbnvvhh	Hdunjuaduqgo	12. 11. 1985 0:00:00	00:07:00.0058894	18	5	5Y1
Jnveunzpmrus	Kcrmeyrfoxzf	29. 8. 1975 0:00:00	03:19:35.6255742	19	5	6OJ
Whgplzhalbju	Sxgtglstflgt	5. 8. 1974 0:00:00	09:57:29.1755145	19	5	6OJ
Drcpmexlgagl	Ppbwhsmovbsc	4. 7. 1989 0:00:00	09:10:25.5597283	18	5	5Y1
Sdowmjbssmzm	Sjgovcysuuyk	15. 8. 1998 0:00:00	06:46:20.7351598	19	5	6IV
Yzyvnfenuhjk	Aaolxhhpxjxv	25. 8. 1965 0:00:00	22:15:44.2785472	21	5	6XJ
Jfxpodzsnebq	Lnkwcttjasgm	16. 1. 1978 0:00:00	06:02:36.2691283	18	5	6BV

1 2 3 4 5 6 7 8 9 10 ...

Obrázek 13: Náhled scénáře 2 pro vyhledání dat z DB zákroků veterinární stanice dle definovaných podmínek.

Dalším zajímavým a zároveň využitelným scénářem nad DB poruch v elektrických sítích je vyhledání poruch, které trvaly určitou dobu. Ve scénáři na obrázku 14 je znázorněn výpis těch záznamů, u kterých porucha trvala kratší dobu než 10 minut (postupné kroky jsou značeny na formulářích F1 a F3 v kapitole 4.2).

Poslední scénář (obrázek 15), který zde bude ukázán, zobrazuje výpis počtu zákroků provedených v lednu 1997 (postupné kroky jsou značeny na formulářích F1 a F2 v kapitole 4.2). Tento scénář tedy pracuje s DB zákroků veterinární stanice.

Obrázek 14: Náhled scénáře 3 pro vyhledání dat z DB poruch v elektrických sítích dle trvání poruch.

Obrázek 15: Náhled scénáře 4 pro vyhledání dat z DB zákroků veterinární stanice dle počtu zákroků v lednu 1997.

7 Závěr

Úkolem bakalářské práce bylo navrhnout a implementovat formuláře, které usnadní uživateli práci při sestavování dotazů nad relační databází. Jako nejvhodnější dotazovací jazyk pro vyvíjenou aplikaci byl zvolen grafický dotazovací jazyk MashQL, a to na základě mnoha vlastností, ze kterých se jako nejpodstatnější jevila schopnost poskytnout grafické uživatelské rozhraní. Samotné řešení se však od jazyka MashQL liší, neboť převádí dotazy posílané do databáze do textového dotazovacího jazyka SQL nad SQL Serverem od firmy Microsoft. Systém je vytvořen jako webová aplikace v jazyce c#, v technologii ASP.NET nazývané WebForms.

Vytvořená aplikace pracuje tak, že uživatel kliká na jednotlivá pole ve formulářích, aby vytvořil požadovaný dotaz pro výběr dat z databáze. Uživatel tedy nepotřebuje speciální znalosti z oblasti dotazovacích jazyků pro definování potřebného dotazu. Dotaz za něj totiž sestaví samotný informační systém.

Původním využitím vzniklé aplikace byla její integrace do již vytvořeného informačního systému pro analýzu dat poruch v elektrických sítích na fakultě informatiky, avšak v průběhu jejího vývoje došlo k zobecnění datové vrstvy k použití pro libovolnou databázi se schématem typu hvězda. Na základě této skutečnosti byl tedy vyvinutý informační systém otestován i pro databázi zákroků veterinární stanice.

Literatura

[1] SINGH, S.K. *Database Systems Concepts, Design and Applications*. E Rutherford: Prentice Hall [Imprint], 2009. ISBN 978-817-7585-674.

[2] POKORNÝ, Jaroslav a Ivan HALAŠKA. *Databázové systémy*. Vyd. 2. přeprac. Praha: Vydavatelství ČVUT, 2003. ISBN 80-010-2789-9.

[3] *Databázové systémy* [online]. Ostrava: VŠB – TUO, 2015 [cit. 2018-02-15]. Dostupné z: http://dbedu.cs.vsb.cz/SubPages/OpenFile.aspx?file=book/dbcb.pdf

[4] *Relational Model of Data Large Shared Data Banks* [online]. San Jose, California: IBM Research Laboratory, 1970 [cit. 2017-11-25]. Dostupné z: https://cs.uwaterloo.ca/ david/cs848s14/codd-relational.pdf

[5] POKORNÝ, Jaroslav. *Dotazovací jazyky*. Praha: Karolinum, 2002. Učební texty Univerzity Karlovy v Praze. ISBN 80-246-0497-3.

[6] *Databázové modely a jazyky* [online]. San Francisco: Wikipedie, 2015 [cit. 2018-02-10]. Dostupné z: http://wiki.matfyz.cz/index.php?title=Datab%C3%A1zov%C3%A9_modely_a_jazyky

[7] *SEQUEL: A Structured English Query Language* [online]. San Jose, California: IBM Research Laboratory, 1974 [cit. 2017-10-12]. Dostupné z: http://www.almaden.ibm.com/cs/people/chamberlin/sequel-1974.pdf

[8] *Jazyk SQL* [online]. Praha: Ondřej Čečák, 2008 [cit. 2017-11-08]. Dostupné z: http://www.cecak.cz/fel/dba/referaty/histori_aplikace_a_vyvoj_jazyka_sql

[9] ULLMAN, Jeffrey D. *Principles of database and knowledge-base systems*. Rockville, Md.: Computer Science Press, 1989. ISBN 08-817-5188-X.

[10] *QUEL Reference Guide* [online]. Palo Alto, California: Actian Corporation, 2012 [cit. 2017-10-11]. Dostupné z: http://docs.huihoo.com/ingres/10s/pdf/quel-reference-guide.pdf

[11] EDITED BY JOACHIM W. SCHMIDT a MICHAEL L. BRODIE. *Relational database systems: analysis and comparison*. S.l.: Springer, 2012. ISBN 978-364-2688-492.

[12] *PIQUE : a relational query language without relations* [online]. Portland: OHSU Digital Commons, 1987 [cit. 2017-12-03]. Dostupné z: https://digitalcommons.ohsu.edu/cgi/viewcontent.cgi?referer=&httpsredir=1&article=1151&context=csetech

[13] *Maier, Chapter 15: Relational Query Languages* [online]. Portland: Dr. David Maier, 1983 [cit. 2017-10-28]. Dostupné z: http://www.dbis.informatik.huberlin.de/fileadmin/research/papers/books/Datenbankbuch_Maier/C15.pdf

[14] *Relational Database Models* [online]. Coventry: Meurig Beynon, 2005 [cit. 2018-01-13]. Dostupné z: http://www.dcs.warwick.ac.uk/people/academic/Meurig.Beynon/CS319/pdf/RelMod.pdf

[15] *Encyclopedia of Database Systems* [online]. Rome: Tiziana Catarci, 2009 [cit. 2018-04-23]. Dostupné z: https://link.springer.com/referenceworkentry/10.1007%2F978-0-387-39940-9_448

[16] *QBE keyword reference* [online]. New York: IBM [cit. 2017-11-10]. Dostupné z: https://www.ibm.com/support/knowledgecenter/en/SS9UMF_10.1.0/ugr/tpc/dsq_qbe_keyboard_ref.html#dsq_qbe_keyboard_ref___i

[17] *QBD*: A Graphical Query Language with Recursion* [online]. New Jersey: IEEE, 1990 [cit. 2018-04-23]. Dostupné z: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=60295

[18] *MashQL: A Query-by-Diagram Language* [online]. Kypr: UNIVERSITY OF CYPRUS, 2008 [cit. 2018-02-13]. Dostupné z: http://www.cs.ucy.ac.cy/ mjarrar/JD08.pdf

[19] *SPARQL Query Language for RDF* [online]. Massachusetts: W3C, 2008 [cit. 2018-04-01]. Dostupné z: https://www.w3.org/TR/rdf-sparql-query/

[20] *W3C* [online]. Massachusetts: W3C [cit. 2018-04-05]. Dostupné z: https://www.w3.org/

[21] *Resource Description Framework (RDF)* [online]. Massachusetts: W3C, 2014 [cit. 2018-04-10]. Dostupné z: https://www.w3.org/RDF/

[22] *Uniform Resource Identifiers (URI): Generic Syntax* [online]. Virginie: The Internet Society, 1998 [cit. 2018-03-18]. Dostupné z: http://delivery.acm.org/10.1145/rfc_fulltext/RFC2396/rfc2396.txt?ip=158.196.195.203&id=RFC2396&acc=ACTIVE%20SERVICE&key=D6C3EEB3AD96C931%2E97A769A0015204DC%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35&___acm___=1524603444_887f2d41d3487ed2d555445b1c3d33c9

[23] *The effect of visual query languages on the improvement of information retrieval skills* [online]. New York: Elsevier, 2010 [cit. 2017-10-20]. Dostupné z: https://www.sciencedirect.com/science/article/pii/S1877042810001308

[24] *Diagrammatic Vs Textual Query Languages: A Comparative Experiment* [online]. Řím: Univerzita v Římě [cit. 2018-04-23]. Dostupné z: https://link.springer.com/content/pdf/10.1007%2F978-0-387-34905-3_5.pdf

[25] KIMBALL, Ralph a Margy ROSS. *The data warehouse toolkit: the definitive guide to dimensional modeling*. Third edition. Indianapolis, IN: John Wiley & Sons, 2013. ISBN 978-1-118-53080-1.

[26] Gono, R., Kratky, M., Rusek, S. *Analysis of distribution network failure databases* (2010) Przeglad Elektrotechniczny, 86 (8)

[27] *Databázový systém pro správu veterinární stanice* [online]. Zlín: Univerzita Tomáše Bati ve Zlíně, 2006 [cit. 2018-02-01]. Dostupné z: http://digilib.k.utb.cz/bitstream/handle/10563/908/habrovansk%C3%BD_2006_bp.pdf ?sequence=1

[28] SHNEIDERMAN, Ben. a Catherine. PLAISANT. *Designing the user interface: strategies for effective human-computer interaction.* 5th ed. Boston: Addison-Wesley, c2010. ISBN 978-0321537355.

[29] *ASP.NET* [online]. Washington: Microsoft [cit. 2018-03-10]. Dostupné z: https://docs.microsoft.com/cs-cz/aspnet/

[30] *ASP.NET – webové formuláře* [online]. Washington: Microsoft, 2011 [cit. 2018-03-12]. Dostupné z: https://docs.microsoft.com/cs-cz/aspnet/web-forms/

[31] *Hypertext Transfer Protocol – HTTP/1.1* [online]. Virginie: The Internet Society, 1999 [cit. 2018-02-25]. Dostupné z: https://www.w3.org/Protocols/rfc2616/rfc2616.html

[32] *Microsoft* [online]. Washington: Microsoft [cit. 2018-04-15]. Dostupné z: https://www.microsoft.com/cs-cz

[33] *Struktura složek webového projektu ASP.NET* [online]. Washington: Microsoft [cit. 2018-04-16]. Dostupné z: https://msdn.microsoft.com/cs-cz/library/ex526337(v=vs.100).aspx

A Datový nosič DVD s aplikací

Na datovém nosiči se nachází aplikace pro uživatelsky přívětivé dotazování.

Struktura aplikace:

- adresář *BC_prace* obsahuje:
  - adresář *.vs* - obsahuje konfigurační data aplikace
  - adresář *BC_projekt* zahrnuje:
    - všechny používané třídy aplikace
    - konfigurační soubory aplikace
    - adresář *App_GlobalResources* - obsahuje slovníky pro každý jazyk aplikace
    - adresář *App_Data* - obsahuje vstupní data aplikace, tedy skripty pro vytvoření DB zákroků veterinární stanice i DB poruch v elektrických sítích
    - ostatní adresáře (např. *App_Start*, *Scripts*)
  - adresář *packages* - obsahuje balíčky nutné pro běh aplikace
  - *BC_prace.sln* - spustitelná aplikace